University of Zurich[UZH]

Analysis of Weather Data using Graph-based and Neural Network Methods

BSc Thesis January 31, 2018

Roland Schläfli
of Bern BE, Switzerland

Student-ID: 12-932-398
rolandschlaefli@gmail.com

Advisor: **Bibek Paudel**

Prof. Abraham Bernstein, PhD
Institut für Informatik
Universität Zürich
http://www.ifi.uzh.ch/ddis

Acknowledgements

First and foremost, I would like to thank Prof. Bernstein for the opportunity to work on such an interesting problem at his research group.

I am also very grateful for the tremendous support provided by my advisor Bibek Paudel, who was always ready to provide constructive feedback and new ideas. His support made the overall creation of this work a thoroughly enjoyable and interesting process.

I would further like to thank Veronika Stolbova for her willingness to consult with us on our first approaches and results, which has helped us identify some early flaws in our implementation. My thanks also go out to Anja Zgraggen and Pascal Zehnder, who have helped improve the formal and content-related correctness of this work.

Zusammenfassung

Der Indische Sommermonsun ist von grosser Bedeutung für über eine Milliarde Menschen. Die Wichtigkeit einer präzisen statistischen Auswertung des Monsunverhaltens wird dadurch verdeutlicht. In dieser Arbeit analysieren wir die geographische Verteilung von extremem Monsunregenfall und entwickeln eine neue Methode zur Vorhersage des Monsunbeginns. Wir berechnen Netzwerke aus korrelierten Orten auf dem Indischen Subkontinent und analysieren diese mit verbreiteten Indikatoren der Netzwerkzentralität. Dadurch zeigt sich eine relative Wichtigkeit von Regionen wie dem Indischen Ozean, dem Tibetischen Plateau und Nordpakistan. Basierend auf aktuellen Methoden im Bereich der Neuronalen Netzwerke entwickeln wir ausserdem ein Modell zur Vorhersage des Monsunbeginns basierend auf räumlich-zeitlichen Wetterdaten. Mit Experimenten zeigen wir, dass unser Modell den Monsunanfang mehrere Tage im Voraus genauer als bestehende Methoden vorhersagen kann.

Abstract

Each year, the Indian Summer Monsoon affects more than one billion people, making clear the importance of accurate statistical analysis of its behavior. In this work, we analyze the spatial distribution of extreme monsoon rainfall and propose a new way of predicting monsoon onset dates. We build networks of correlated locations on the Indian subcontinent, analyzing them with established centrality measures. These measures reveal the relative importance of locations like the Indian Ocean, the Tibetan Plateau, and Northern Pakistan. We additionally adopt recent advances in the area of neural networks to predict monsoon onset dates based on spatiotemporal meteorological datasets. With experiments on these datasets, we show that our model is able to predict onset dates more accurately than existing methods several days in advance.

Table of Contents

Abbreviations

BoB Bay of Bengal

CNN Convolutional Neural Network

E1-E4 Models based on ERA-Interim

EASM East Asian Summer Monsoon

ECMWF European Centre for Medium-Range Weather Forecasts

EEG Electroencephalogram

ENSO El Niño Southern Oscillation

EV01-EV16 Evaluation Schemes

GDP Gross Domestic Product

GPM Global Precipitation Measurement (Mission)

IMD India Meteorological Department

ISM Indian Summer Monsoon

ITCZ Intertropical Convergence Zone

LSTM Long-Short Term Memory

MAE Mean-Absolute Error

MAM March-April-May

MoK Monsoon over Kerala

MSE Mean-Squared Error

NASA National Aeronautics and Space Administration of the USA

NP North Pakistan

JAXA Japanese Aerospace Exploration Agency

JJAS June-July-August-September

OND October-November-December

RMSE Root Mean-Squared Error

RNN Recurrent Neural Network

T1-T5 Models based on TRMM

TRMM Tropical Rainfall Measurement Mission

TP Tibetan Plateau

UTC Coordinated Universal Time

WG Western Ghats

WMO World Meteorological Organization

WNPSM Western North Pacific Summer Monsoon

1

Introduction

The Indian Summer Monsoon (hereafter also referred to as ISM[1]) is one of the most significant meteorological events on our planet. Each year during the time-frame from June to September, it shapes the lives of the more than one billion people living on the Indian subcontinent. Stretching from the Tibetan Plateau and the Himalayas up north down to the coasts of the Indian Ocean, the Indian subcontinent is an enormous geographical area that encompasses countries like India, Pakistan and Bangladesh, amongst others. The terms "Indian subcontinent" and "South Asia" are often used interchangeably, even though they are both only loosely defined.

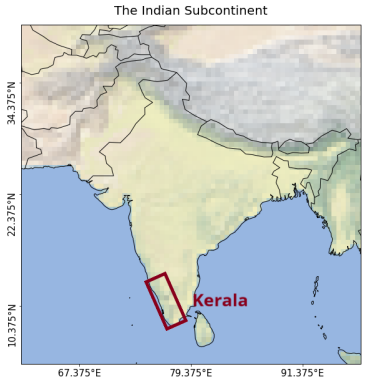

Figure 1.1: Geographical area of the Indian subcontinent as used in our work. Kerala, a region especially important for the prediction of ISM onset, is emphasized in red.

As illustrated in Figure 1.1, India makes up for a major portion of the Indian subcontinent (hence the matching naming). Due to it being one of the most highly populated countries on earth, as well as one of the first to be impacted by the yearly occurrence of the ISM, we focus our work mainly on the ramifications of the ISM on India and its general population.

The ISM is of great importance to the Indian population: many parts of India receive up to 80% of all rainfall during the monsoon season, and significant parts of the population obtain their drinking water from monsoon rainfalls (Stolbova, 2015). Farmers further depend on the arrival of these rainfalls to be able to water their crops and feed their livestock. The monsoon is of similarly high impact for the Indian economy at large, as the agricultural sector makes up almost one-fifth of India's gross-domestic-product (GDP) (Central Intelligence Agency, 05.01.2018).

In itself, the ISM is a highly complex climatological phenomenon. It displays considerable variability in both timing and strength and is influenced by many factors on regional and global

[1] An overview over all abbreviations used in this work can be found in the list of abbreviations (front matter *xii*).

scales. Due to its evident importance, the behavior of the ISM has been a focus of climate researchers for many decades, bringing to light many theories and hypotheses. Numerous approaches for the prediction of monsoon onset and withdrawal, the times at which monsoon weather conditions reach and leave a particular location, have been proposed over the years. Other research seeks to analyze and predict the amount of rainfall that locations in India get during the monsoon season, which could allow the population to better prepare for the potential dangers of extreme rainfall.

Many factors that drive the ISM are still unknown or unexplained, even after many years of research. Prediction tasks tend to be hard to accomplish and rely on sophisticated climatological models, while their accuracy is often subpar when compared with weather prediction in "more predictable" countries in Europe or North America. The India Meteorological Department (IMD), the agency that is officially tasked with the analysis and prediction of ISM behavior, predicts the monsoon from up to a one month lead and manages to do so with a root-mean-squared error (RMSE) of about four days (Pradhan et al., 2017). However, it has regularly failed to predict even drought-like conditions (Paliwal, 24.09.2017).

A problem in prediction tasks is often the computing power that is needed to drive accurate models for complex phenomena. More data than ever before is being collected by satellites and sensors every day, resulting in high computational requirements. Additionally, climatological phenomena are ever-changing and are subject to large-scale trends like climate change, making successful predictions even harder to accomplish. Recent climatic trends tend to further increase the variability of natural phenomena like rainfall, resulting in increased numbers of droughts as well as extreme rainfall events in India. The tremendous impact of the climate on the welfare of society and the economy at large makes having the capability to predict weather events accurately, be it rainfall, drought or the monsoon onset, more crucial than ever.

The availability and affordability of computing resources have been increasing at a very fast pace during the last decade, leading to the emergence of many sophisticated methods for data analysis and prediction. Neural networks, a category of machine learning models that has its origins in the early 20th century, have only become practical during recent years, as the amount of data and computational power required to train them adequately had never been available before. Since the reemergence of neural networks, they have become a key component in complicated machine learning tasks like image recognition and natural language processing. As they even seem to be able to learn patterns in data that humans do not know of yet, they could be a good contender for tasks like the prediction of ISM behavior. However, the application of neural networks and deep learning in this climatological domain is not yet as prevalent as it is in classical computer science domains.

In this work, we seek to evaluate the applicability of the aforementioned neural networks to the domain of ISM onset prediction. Also, we analyze the spatial and temporal distribution of extreme rainfall during the different seasons of monsoon. Because of the complex behavior of the ISM, we dedicate Chapter 2 of this work to a short summarization of research about the inner workings of the ISM. We explain the major factors that are known to influence monsoon and try to further motivate the significance and social impact of the monsoon for the people in India and its surrounding countries.

The remainder of this work is structured into two major parts: the first part (Chapter 3) is dedicated to the analysis of extreme rainfall events before, during and after the monsoon season. Such events are regularly seen all over India and represent one of the most significant challenges the local population faces from the ISM. Extreme rainfall can cause enormous damage by flooding cities or destruction of essential infrastructure. As such, there is a clear need to

analyze and predict the spatial and temporal distribution of these extreme rainfall events.

Basing our first part on the work of Stolbova (2015), we closely follow her methodology and choice of dataset: all work in Chapter 3 is based on the Tropical Rainfall Measurement Mission (TRMM) dataset, a precipitation dataset compiled by the National Aeronautics and Space Agency (NASA) and the Japanese Aerospace Exploration Agency (JAXA). The TRMM product we use in this work contains data for the years 1998-2016 at a 0.25° resolution. Because of its high resolution, the TRMM dataset is a popular choice for climate research whenever a high level of detail is needed.

As a first step in the analysis of the aforementioned TRMM dataset, we extract extreme rainfall events and build event series for different locations in India. We then calculate the synchronicity of different pairs of locations[2] and build a "climate network" connecting only the most significantly synchronous locations.

Analysis of the resulting network using network centrality measures finally gives us an intuition about the importance of a location in the network. In this work, we use several different measures: firstly, we calculate the degree of locations, giving us the number of locations to which a location is synchronous. Secondly, the betweenness of a location gives us the number of shortest paths between locations that this location lies on. And finally, the PageRank algorithm, which has been created by Larry Page and Sergey Brin to calculate the importance of web pages in their search engine Google. Knowledge about these measures could potentially help in the analysis or even prediction of monsoon behavior. For example, extreme rainfall in a location that is very central might be a reason to implement safety measures in connected locations.

Our climate network analyses show interesting patterns over several expected and unexpected parts of India: the parts of the Indian ocean on either side of India are very central during the pre-monsoon season, while the monsoon season is dominated by patterns over Northern Pakistan. The post-monsoon season exhibits strong centrality over central India. Additionally, the Tibetan Plateau is one of the most important areas over all seasons, especially when ranked by the PageRank algorithm. While most of the observed patterns correlate well with the findings of Stolbova (2015), we also find some significant differences mainly in the PageRank analysis and during the post-monsoon season.

The second part of this work (Chapter 4) deals with the critical issue of monsoon onset prediction. The onset date for a location in India depicts the point at which the monsoon reaches said location with strength and durability above a predefined threshold. We try to predict such an onset date for the Kerala region on the southern tip of the Indian subcontinent (as shown in Fig. 1.1), which is one of the first locations reached by the ISM and as such marks the beginning of monsoon for the whole Indian subcontinent.

We base our onset prediction task on two climate datasets: in addition to the TRMM dataset we have already introduced, we use the ERA-Interim dataset as produced by the European Centre for Medium-Range Weather Forecasts (ECMWF). Starting in 1979, ERA-Interim is available for a longer period than TRMM but at a much less detailed 0.75° resolution. In contrast to the TRMM dataset, which is purely focused on the amount of precipitation, ERA-Interim contains many different features (e.g., temperature) at various vertical levels (measured according to their pressure).

Based on the datasets as previously mentioned, we build neural network architectures using the Python Keras 2.0 and Tensorflow 1.4 libraries. Experiments with each model allow us to

[2]Simplified example: two locations are synchronous if an extreme event in one location tends to be followed by an extreme event in the other location.

evaluate and shortly summarize our most important findings. We iteratively improve upon the findings of each model, building models with vastly different architectures as well as for a number features and combinations thereof (TRMM, ERA-Interim, different sets of onset dates). We first show the model that has been found to exhibit the overall best learning capabilities and evaluate it in detail, following up with an overview of the models that have been evaluated beforehand.

The experiments and evaluations based on our final model architecture regularly exhibit good prediction performance: some model architectures manage to predict the monsoon onset from one to thirty days in advance while doing so with less than three days of mean-absolute error (MAE). If we were to predict the monsoon onset over Kerala on the 10th of May, our best model would be able to predict the test set with less than three days of MAE and within a 95% confidence interval ranging from around two to four days of MAE. Given these results, our model outperforms several existing methods based on these datasets and should provide a good foundation for more advanced onset neural network prediction models based on (possibly other) spatiotemporal datasets.

Finally concluding our work in Chapter 5, we summarize our overall findings and reflect on the results of our climate network analyses in Chapter 3 and the onset prediction task in Chapter 4. We additionally provide ideas for the improvement of these problems and possible future research based on the topics of extreme event synchronization and monsoon onset prediction using deep learning methods.

2

Overview of the Indian Summer Monsoon

The Indian Summer Monsoon is a global climatic phenomenon that has been actively researched for decades. Even though many theories and hypotheses exist, the factors influencing its strength, timing and variability have still not been fully identified and understood. This chapter aims to provide a short overview of the observed behavior and current scientific understanding of ISM, as far as the foreknowledge might prove useful for the further parts of this work.

2.1 Seasons & Progression

The ISM progresses over the Indian subcontinent in three main phases: the pre-monsoon season (March-May), the primary monsoon season (June-September) and the post-monsoon season (October-December) (Stolbova, 2015).

The months of the pre-monsoon season correspond to the Indian summer and are characterized by high temperatures with low amounts of rainfall over regions in the Himalayas and south-west India. Many predictions about the further development of ISM are commonly based on the pre-monsoon season, especially on the patterns that occur leading up to the primary monsoon season (Stolbova, 2015).

A sudden transition with rapidly increasing amounts of daily rainfall, moisture, and kinetic energy, as well as increased vertical shear in horizontal winds[1], then marks the beginning of the primary monsoon season (Pradhan et al., 2017). This transition, the "onset" of monsoon, occurs at different times for different parts of the Indian subcontinent, as can be seen in Fig. 2.1. First arriving in Kerala on the southwestern tip of the subcontinent around the first of June, the ISM then propagates north- and northwestwards, where it reaches the Pakistani border around the 15th of July (Willetts et al., 2017).

After the ISM has fully covered the Indian subcontinent in July, most places experience its weather conditions well into September. When the monsoon starts its retreat from northern India, its progression reverses, and it gradually withdraws from the subcontinent, entirely leaving the subcontinent in October.

During the post-monsoon season, parts of the Indian subcontinent experience a second extended period of rainfall called the *North-East Monsoon*. Some places in north-eastern India receive more rainfall during the North-East Monsoon than during the primary season of the ISM. This work is, however, focused mostly on the Indian Summer Monsoon, because of which we will not be covering the North-East Monsoon in more detail.

[1] A gradient between wind speeds at different heights or pressure levels.

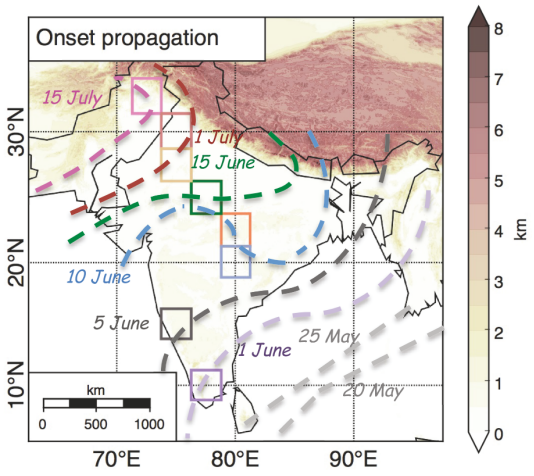

Figure 2.1: Propagation of monsoon over the Indian subcontinent as depicted in Stolbova (2015), based on a recreation of the official figure by the IMD.

2.2 Main Drivers

In its most basic form, the ISM can be seen as a sea breeze of enormous extent. The hot summer weather during the pre-monsoon season causes a heating of the Indian subcontinent. An increasingly large temperature gradient due to slower heating of the Indian Ocean then causes air to flow onto the subcontinent, bringing with it the moisture necessary to cause precipitation (Willetts et al., 2017).

Intense heating and heat dissipation of the high-altitude Tibetan Plateau lead to an increase in the tropospheric temperature, creating an area of low-pressure near the surface that attracts moisture from surrounding areas over the Indian Ocean (Stolbova, 2015). The low-pressure zone further causes strong vertical air currents from south of the Tibetan Plateau, aiding the ISM with its propagation into the far north of India (Pradhan et al., 2017).

On a larger scale, the ISM has been linked to several parts of the global atmospheric circulatory system. A shift of the subtropical jet stream to the north of the Tibetan Plateau as well as interaction with the westerly jet stream have been found to influence the behavior of the ISM (Ordoñez, Gallego, Ribera, Peña-Ortiz, & García-Herrera, 2016; Stolbova, 2015). A strong westerly jet in the lower troposphere over Kerala seems to correlate with the monsoon onset of the region (Ordoñez et al., 2016). Additionally, the Somali jet passing over the Arabian sea cools down the body of water, further strengthening the temperature gradient (Stolbova, 2015).

The trade winds of the northern and southern hemisphere meet to create the Intertropical Convergence Zone (ITCZ), a belt of low-pressure close to the thermal equator. When the ITCZ moves north following the summer months, it further increases the variability of weather events and thus reduces their predictability (Stolbova, 2015).

A further factor in the development and strength of the ISM is the El Niño Southern Oscillation (ENSO). The warming of the Humboldt ocean current during El Niño years causes the land-ocean temperature gradient to decrease. This tends to decrease the amount of rainfall the Indian subcontinent receives and can delay the onset of monsoon by several days (Pradhan et al., 2017; Willetts et al., 2017). Conversely, the cooling of the current during La Niña can result in more rainfall and an earlier onset due to a higher temperature gradient.

A combination of the factors above creates a low-pressure channel (a "monsoon trough") closely south of and parallel to the Tibetan Plateau that serves as a primary source of moisture during the ISM (Stolbova, 2015).

The World Meteorological Organization (WMO) describes the ISM as only part of a global monsoon system, by which it is interlinked with other monsoon regions all over the world (World Meteorological Organization, 2005). The ISM as described is part of a larger Asian monsoon system, which additionally includes the *East Asian Summer Monsoon (EASM)* and the *Western North Pacific Summer Monsoon (WNPSM)*, as is shown in Fig. 2.2 (Yihui & Chan, 2005). While the ISM has been the focus of monsoon research for decades, the theory of a global circulatory monsoon system has been investigated for a much shorter period (World Meteorological Organization, 2005). As such, much about the global teleconnections between different regional monsoon systems is still unknown.

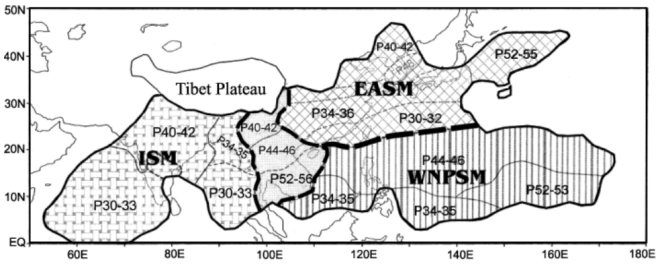

Figure 2.2: Monsoon system over Asia as found in Yihui and Chan (2005).

This summarization of the impacting factors of the ISM is not exhaustive, as the ISM is much more complex and some of its behavior and teleconnections have still not been fully explained. However, knowledge of these fundamental factors should already provide a good intuition for Chapter 3 and Chapter 4.

2.3 Trends

Much like the global climate, the ISM is exposed to trends that can impact its variability and predictability. A major trend that was prevalent during the second half of the 20th century was a decrease in monsoon rainfalls over northern-central India close to the Tibetan Plateau (Jin & Wang, 2017). One theory states that such a drying trend could be connected to a warming of the surrounding Indian Ocean. A resulting weakening of the land-ocean temperature gradient could have been responsible for the lower rainfall amounts. A further hypothesis states that large-scale deforestation could have decreased the amount of transpiration from plants, resulting in less moisture and thus precipitation (Jin & Wang, 2017).

In addition to the weakening precipitation, it has been found that the frequency of heavy and extreme rainfall events increased by 50% and 100% respectively. At the same time, short and long dry spells were found to occur increasingly often (Auffhammer, Ramanathan, & Vincent, 2012). The drought during the 2009 monsoon season was more severe than most that occurred in the previous decades (Auffhammer et al., 2012). As such, the overall trend seems to be an increase in general extremes, be it extreme rainfall or no rainfall at all.

Surprisingly, Jin and Wang (2017) have found that the precipitation weakening trend has reversed for most parts of India after 2002, which is also shown in Fig. 2.3. The reversal is

attributed to changing trends of features like the temperature gradient between land and sea temperature during the pre-monsoon season, which, since 2002, is increasing at both a surface level and in the mid-troposphere, while it was decreasing beforehand (Jin & Wang, 2017).

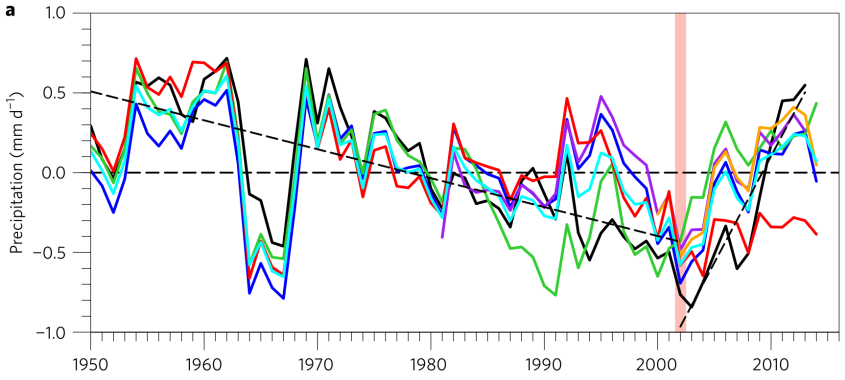

Figure 2.3: Trends of various precipitation datasets as depicted in Jin and Wang (2017). The TRMM dataset is represented by the orange line.

2.4 Social Impact

As we have already seen, the ISM is one of the most impactful large-scale meteorological events on earth. It affects the lives of up to one-fourth of the world's population: the people living on and around the Indian subcontinent (Stolbova, 2015). The heavy rainfalls that the ISM brings with it are of great importance for the population in India and many of its surrounding countries.

Livelihood on the Indian subcontinent is strongly coupled to a timely occurrence of the ISM. Monsoon rainfalls are responsible for 80 percent of the annual precipitation on the Indian subcontinent (Jin & Wang, 2017). Farmers depend on these rainfalls to water their crops and feed their livestock. Up to 2012, more than half of a year's rice harvest was still grown during the monsoon period (Auffhammer et al., 2012). The agricultural sector accounts for almost one-fifth of India's GDP, and about 50% of the population in India either directly or indirectly depend on the agricultural sector (Central Intelligence Agency, 05.01.2018), making the ISM equally important for the economy as a whole. Historically, years with sub-par monsoon rainfalls or even droughts have caused massive losses for farmers as well as high reductions in the general economy and welfare. For example, the heavy drought in 2009 caused the rice harvest to decrease by 14% (Auffhammer et al., 2012).

The apparent impactfulness of the ISM makes clear the need of being able to predict the monsoon behavior and its onset and withdrawal dates accurately. If farmers knew in advance when monsoon rainfalls would start and how long they would last, they could wait until the appropriate time to plant their crops. On the contrary, if they get surprised by either early or late onset of monsoon or even drought-like conditions, their crop yield might be reduced, or their crops destroyed entirely. Reliable predictions are thus highly called for, as they could reduce these risks as well as general concerns about food security. Furthermore, with the increasing trend in extreme events on both sides of the scale, analyzing their spatial distribution poses an interesting research question. It is probable that the occurrence of extreme events adheres to some patterns, the understanding of which could help mitigate risks of floods or landslides.

3

Part 1: Synchronization of Extreme Events

During the primary monsoon season, floodings and landslides caused by extreme rainfall can lead to massive societal and environmental damage. As we have already seen, it is crucial to be able to analyze, detect and potentially predict such extreme rainfall events, especially as their proportion tends to further increase with global warming (Stolbova, 2015).

One recently proposed way for such analysis is the computation of climate networks, followed by an analysis of the resulting networks using well-known network centrality measures (Stolbova, 2015). The approach taken in Stolbova (2015) has its foundations on the work of Quian Quiroga, Kreuz, and Grassberger (2002), which is why we first go over their approach to event synchronization (Section 3.2.1). We then summarize the principles of climate networks as applied by Stolbova (2015) in Section 3.2.2, after which we go on to explain the network measures that will be applied to the obtained climate networks (Section 3.2.3). Before we go into any of this related work, Section 3.1 first introduces the TRMM dataset that we will use in this chapter.

After going over the foundational related work, Section 3.3 provides an overview of our approach to the creation of climate networks. Following up in Section 3.4, we analyze and visualize the computed climate networks using centrality measures and try to interpret the results in contrast to the known factors of monsoon behavior. We further compare our results to the ones obtained in Stolbova (2015) and assess any apparent differences. Concluding this chapter in Section 3.5, we elaborate the usefulness of our results and how they could be improved upon.

3.1 The TRMM Dataset

The TRMM dataset is a precipitation research effort by NASA and JAXA. It is based on the TRMM observatory, a satellite that was launched into space on the 27th of November, 1997. The products based on TRMM range from the raw output of the multitude of sensors on the satellite to the highly aggregated and gridded rainfall estimates we will be using in this work (Goddard Earth Science Data Information and Services Center, 2016).

More specifically, the TRMM product that we will be using is a 3-hourly estimate of surface rainfall aggregated from the satellite sensors in combination with surface gauge values and imagery from other satellites. This product is referred to as 3B42 or TMPA and is also available in a daily variation, where the eight 3-hourly measurements (i.e., 00:00, 03:00, 06:00 and so on) have been summed up to provide a single daily rainfall estimate. We use this daily TMPA product during the remainder of this work and, for simplicity, generally refer to it as TRMM.

TMPA is available for the area between 50°N. and 50°S. We subset this area to cover the entire Indian subcontinent (4.125-40.625°N, 61.125-97.625°E). These border coordinates are a small superset of the ones used in Stolbova (2015): the grid has been extended such that it can be cleanly aggregated from a 0.25° spatial resolution to the 0.75° resolution of the ERA-Interim dataset we use in Chapter 4. The gridded area we extracted from the TRMM dataset is visualized in Fig. 3.1.

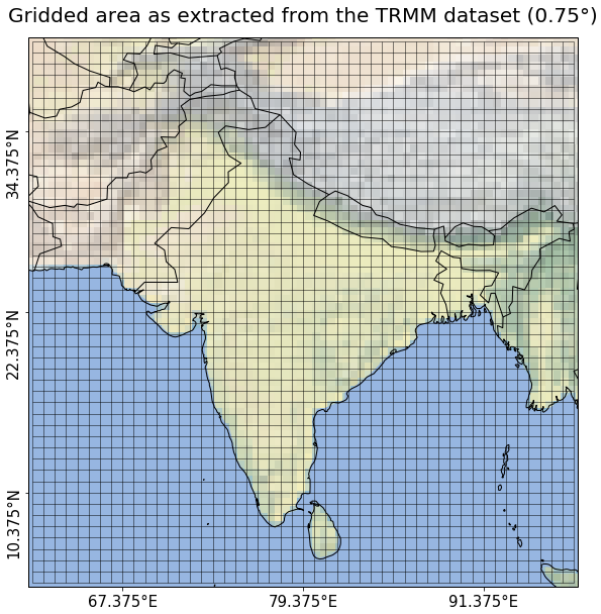

Figure 3.1: Overview of the Indian subcontinent as extracted from the TRMM dataset (4.125-40.625°N, 61.125-97.625°E). The TRMM dataset has been aggregated to match the 0.75° resolution of the ERA-Interim dataset.

The TRMM dataset is unique in that it offers very high-resolution precipitation estimates since January 1998. It can prove useful for research based on the distribution, frequency, and intensity of rainfall, i.e., for calculating extreme rainfall events (Stolbova, 2015). However, it has to be taken into consideration that the TRMM products are based on complex algorithms and have been derived from different sensors and sources (G. J. Huffman, Pendergrass, & National Center for Atmospheric Research Staff, 2017).

After the TRMM satellite's fuel went low in 2014, it was decommissioned in April 2015 and re-entered earth's atmosphere in June 2015. The TMPA product is still being produced until 2018, albeit without the sensors of the TRMM observatory and with less accuracy (G. Huffman & Bolvin, 2017). Built on the success of the TRMM mission, NASA and JAXA have already launched its successor Global Precipitation Measurement (GPM) in 2014 (Goddard Earth Science Data Information and Services Center, 2011). However, as its data is only available starting from 2014, GPM is currently unsuitable for long-term precipitation research. TRMM or an alternative dataset are currently still needed, but the updated algorithm developed for GPM will soon be applied to the existing TRMM data. The availability of reprocessed data back to 1998 can thus be expected in 2018 (G. Huffman, 2016).

3.2 Related Work

The concept of event synchronization applied in Stolbova (2015) was first proposed in Quian Quiroga et al. (2002), where they sought to develop a simple algorithm that could be applied to any two time series of events, resulting in a measure that defines the synchronization of said time series. Synchronization measures are related to standard measures like cross-correlation but can convey more complex relationships due to their non-linearity (Quian Quiroga et al., 2002).

While the work of Quian Quiroga et al. (2002) focuses on the analysis of electroencephalogram (EEG)[1] time series, their event synchronization approach is explicitly applicable to other domains. The works of Malik, Marwan, and Kurths (2010) and Stolbova (2015) both expand upon simple event synchronization. However, they do so in different ways: Malik et al. (2010) apply hierarchical clustering algorithms to identify different regions and their respective characteristics regarding the occurrence of extreme rainfall events. Stolbova (2015) builds "climate networks" and analyzes them with established network measures. For our work, we will focus on the latter approach, using the work of Stolbova (2015) as a foundation for this chapter.

3.2.1 Event synchronization

The basic principle of the event synchronization measure as found in Quian Quiroga et al. (2002) is setup as follows: given the two time series i and j and their respective events l and m occuring at times t_l^i and t_j^y, we measure the synchronicity for all possible pairs of events between both series. We classify events as synchronous if they occur closely simultaneous, i.e., within a certain range from each other. This allowed range of occurrence is called *time lag* and is calculated by taking the minimum interevent distance τ_{lm}^{ij} like so[2]:

$$\tau_{lm}^{ij} = 0.5 * min\left\{t_{l+1}^i - t_l^i, t_l^i - t_{l-1}^i, t_{m+1}^j - t_m^j, t_m^j - t_{m-1}^j\right\} \tag{3.1}$$

The synchronicity J_{ij} of any two events t_i^x and t_j^y is then calculated as follows:

$$J_{ij} = \begin{cases} 1, & \text{if } 0 < t_i^x - t_j^y \leqslant \tau_{ij}, \\ 0.5, & \text{if } t_i^x = t_j^y, \\ 0, & \text{else.} \end{cases} \tag{3.2}$$

Applying this to the full time series i and j, the number of synchronous events where an event in i leads an event in j is defined like:

$$c(i \mid j) = \sum_{l=1}^{s_i} \sum_{m=1}^{s_j} J_{ij} \tag{3.3}$$

The same formula applies to the reversed situation, i.e. $c(j \mid i)$.

[1] A way of measuring brain activity.

[2] If event rates were fixed, a global time lag τ could be defined, greatly simplifying calculations.

Combining the results of $c(i \mid j)$ and $c(j \mid i)$ and normalizing them by the total numbers of events s_i and s_j, the *strength of synchronization* is defined as

$$Q_{ij} = \frac{c(i \mid j) + c(j \mid i)}{\sqrt{(s_i - 2)(s_j - 2)}} \tag{3.4}$$

where $Q_{ij} = 1$ means that the time series are completely synchronized, i.e., that each event in i is either synchronously lead or followed by an event in j.

3.2.2 Climate networks

Looking at the TRMM dataset as shown in Fig. 3.1, each cell of its coordinate grid represents a separate precipitation time series. As the analysis of each monsoon season is performed separately, one such coordinate grid per season needs to be extracted[3].

The time series in the resulting grids are not event-based and thus cannot be directly used to calculate synchronization. They can, however, easily be transformed into event series by extracting only the days where extreme rainfall occurred. As per Stolbova (2015), such days are defined as days with rainfall that exceeds the 90th percentile for the respective location.

The resulting series of extreme events can be used to calculate the synchronicity of different locations (grid cells) in the TRMM dataset, closely following the approach we have already introduced in Section 3.2.1. Computing the synchronicity for all possible permutations of two such locations then results in a synchronization matrix as can be seen in Fig. 3.2.

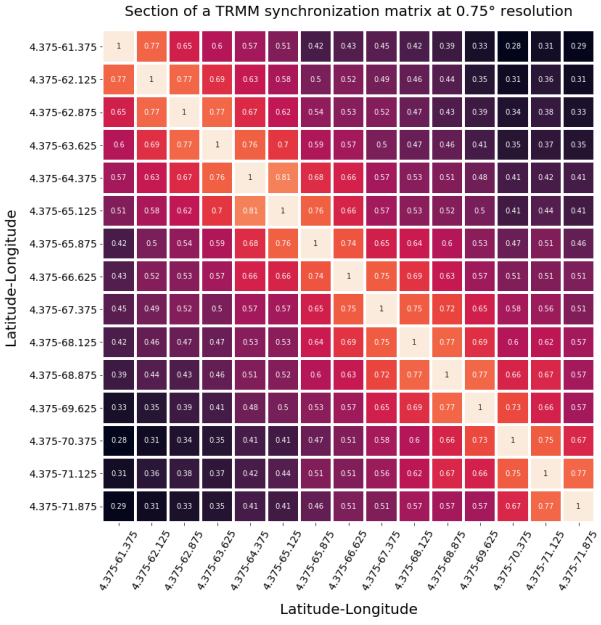

Figure 3.2: Exemplary synchronization matrix for TRMM at a 0.75° resolution. We only show the top left 15x15 section of an actual matrix, as the full dimensions are 2401x2401 for a 0.75° resolution.

[3]The time series are then simply the respective months of each year, concatenated into a single series. For example March 1998, April 1998, May 1998, March 1999, April 1999, and so on.

Applying a numerical threshold to this synchronization matrix results in a matrix that contains only the most significantly synchronous values. Everything else is set to zero, including the diagonal of the matrix, as this would result in loops in the graph later on. Additionally, all elements above the threshold are set to one, yielding the adjacency matrix for an undirected, unweighted network (Fig. 3.3). Stolbova (2015) uses the 95th percentile for the threshold applied, as this removes all but the most statistically significant values.

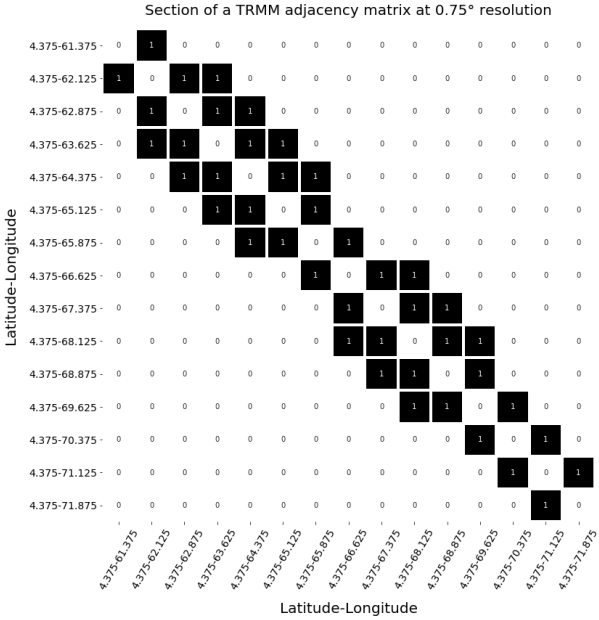

Figure 3.3: Exemplary adjacency matrix for TRMM at a 0.75° resolution. We only show the top left 15x15 section of an actual matrix, as the full dimensions are 2401x2401 for a 0.75° resolution.

Based on the adjacency matrix, a network can then be computed and analyzed. However, before we detail our implementation of it, the next section briefly describes the measures that we will be using for said network analysis.

3.2.3 Network centrality

Constructing climate networks from an adjacency matrix as seen in Section 3.2.2 enables the analysis of their structural features using various network-based measures. We now briefly introduce these measures, before we go on to apply them in the next section.

The climate networks in Stolbova (2015) are analyzed using the *degree* and *betweenness* of nodes as well as using the *average/maximal geographical link length* between them. We also base our analysis on the *degree* and *betweenness* measures but replace the link length calculations with the *PageRank* algorithm, which should also show interesting patterns in our opinion.

Degree

The *degree* of a node in a network is quite simply defined as the number of links that are connected to this node. The same holds for vertices in graphs and their connected edges.

Betweenness

Calculating the *betweenness* or *betweenness centrality* of a node is a more involved effort. A node has a high betweenness if a large portion of shortest paths in the network passes through the respective node. If the resulting betweenness is to be an exact measure, this can necessitate the calculation of all shortest paths in the network, which gets increasingly complex with the size of the network. There are, however, algorithms that achieve both accuracy and speed when calculating betweenness, one of the most popular being the approach proposed by Brandes (2001). This algorithm is also used by the Python library *networkx*, which we will be using for our calculations.

PageRank

The final measure that we will apply to climate networks, the *PageRank* algorithm, was originally developed and published by the founders of Google (amongst others), Larry Page and Sergey Brin, during their studies at Stanford University (Page, Brin, Motwani, & Winograd, 1999). Originally developed as a way to rank indexed websites by their importance, the algorithm has since been applied to a variety of other problem domains (e.g., social network analysis, neuroscience, and many others). The PageRank algorithm seeks to calculate the importance of a node in a network by taking into account its incoming and outgoing links as well as the PageRank and number of outgoing links of the thereby connected nodes (which is intuitively shown in Fig. 3.4).

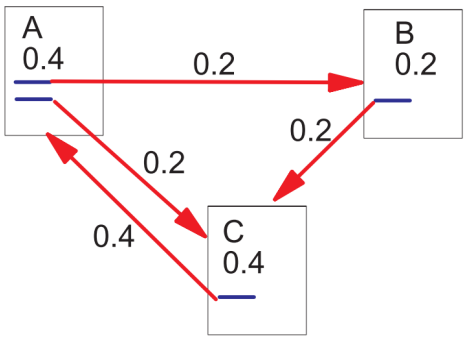

Figure 3.4: The simplified PageRank algorithm as presented in Page et al. (1999).

As intuitively described by Page et al. (1999), PageRank can be thought to model the behavior of a "random surfer" that merely continues clicking links on the current page, traversing the network in a random fashion. The PageRank of a particular page then represents the probability that such random surfers end up accessing the respective page. However, to prevent any random surfer from getting stuck in loops (i.e., pages that link between each other), the simplified PageRank algorithm is further extended by a teleportation capability: a random surfer will jump to a random page with probability α, while following a link to another page with probability $1 - \alpha$.

3.3 Implementation

Our implementation of the event synchronization and climate network computations is strongly based on the concepts and measures as described in Section 3.2.1 and Section 3.2.2. As our algorithmic implementation is most certainly different than the one used in the original work (Stolbova, 2015), we detail our approach in this section.

3.3.1 Calculating event synchronization

Before we can calculate the synchronicity for any two locations, the precipitation time series need to be transformed into series of events. Assume that we have extracted pre-monsoon precipitation time series as shown in Table 3.1 from the TRMM dataset and want to convert these into extreme event series.

Latitude	**Longitude**	**01.03.98**	**...**	**29.05.98**	**30.05.98**	**31.05.98**	**01.03.99**	...
13.375	67.375	0.0	...	0.12	2.31	2.85	0.00	...
16.375	91.375	0.0	...	0.09	34.80	49.49	0.00	...
34.375	67.375	0.52	...	0.00	0.00	0.00	0.00	...
34.375	88.375	0.86	...	2.01	51.85	68.72	0.29	...

Table 3.1: Precipitation time series during pre-monsoon at 4 exemplary locations (TRMM, 0.75°).

We then calculate the 90th percentile for each row and apply the result as a threshold to the respective row. This yields event series where each value represents the date of an extreme event. If applied to the time series in Table 3.1, this results in event series as shown in Table 3.2 .

Latitude	**Longitude**	**1**	**2**	**3**	**4**	**5**	**6**	**...**
13.375	67.375	07.04.98	31.05.98	08.05.99	12.05.99	13.05.99	15.05.99	...
16.375	91.375	17.05.98	18.05.98	19.05.98	30.04.99	01.05.99	05.05.99	...
34.375	67.375	03.03.98	29.03.98	02.04.98	03.04.98	09.04.98	22.04.98	...
34.375	88.375	18.03.98	30.03.98	31.03.98	01.04.98	05.04.98	19.04.98	...

Table 3.2: Events in the pre-monsoon extreme event series at 4 exemplary locations (TRMM, 0.75°).

Going forward, we use *epoch timestamps* to represent dates instead of the full representation shown in the examples of this section, as they are much easier to process using mathematical formulas (but less intuitive to visualize). Epoch timestamps measure the number of seconds (without leap seconds) that have elapsed since the 01.01.1970 at 00:00 UTC. For example, *01.03.1998 00:00* would be represented as *888710400*. An exemplary event series based on epoch timestamps is presented in Table 3.3. Such timestamps or similar numerical representations are internally used in many applications and algorithms, as calculating differences between dates or shifting a date is as simple as performing mathematical addition or subtraction operations.

Event	1	2	3	4	5	6	...
Date	07.04.98	31.05.98	08.05.99	12.05.99	13.05.99	15.05.99	...
Epoch	891907200	896572800	926121600	926467200	926553600	926726400	...

Table 3.3: Exemplary series of events represented with epoch timestamps (TRMM, 0.75°).

Synchronization of two locations

Having explained the necessary setup of the event synchronization algorithm, we now focus on more concrete aspects of our implementation. To calculate the synchronization between any two locations (i.e., *location1* and *location2*), we need to process both time series and compare each event in *location1* to the appropriate events in *location2*, taking the immediate neighbors of both locations into account. We found that this naturally fits the description of a *sliding window* approach: a sliding window of size three over the event series of a location always encompasses a current event, its predecessor, and its successor and thus everything we need for the time lag calculation. Application of a sliding window to *location1* and nesting another such window for *location2* then leads us to the naive implementation shown in Listing 1[4].

The event synchronization algorithm as described in Listing 1 results in the number of synchronous events between *location1* and *location2* where the event in *location1* leads the event in *location2* (i.e., $c(\text{location1} \mid \text{location2})$). The above is, however, only part of the full synchronization calculation, as we also need the number of synchronous events where *location1* leads *location2*, i.e., we need to evaluate $c(\text{location2} \mid \text{location1})$. This is the reason that simultaneous events only count as "half synchronous": they are counted in both these calculations and, in total, then result in one synchronous event.

Calculating the strength of synchronization boils down to a simple application of Eq. (3.4) from Page 12 and is further presented in Listing 2. Notice that, in addition to the synchronization coefficient, we return both numbers of synchronous events (i.e., $c(i \mid j)$ and $c(j \mid i)$). This is used to build a directed network later on.

Computing the values of the synchronization matrix

Given a matrix of extreme events as shown in excerpt in Table 3.2 (but based on epoch timestamps), we want to calculate the synchronization coefficient and the number of synchronous events for all pairs of locations. Table 3.4 shows the state of a preliminary synchronization matrix before running any of the calculations. Locations are always perfectly synchronous to themselves, leading to a diagonal that is always filled with ones (in the case of a synchronization coefficient matrix).

Next to a synchronization matrix that contains all synchronization coefficients (Q_{ij}), we want to generate two more matrices with the same dimensions: one matrix containing the total count of synchronous events for all pairs of locations ($c(i \mid j) + c(j \mid i)$) as well as a matrix that separately contains the results of $c(i \mid j)$ and $c(j \mid i)$ (and is thus asymmetrical). We can generate all three of these matrices in a single pass, as $c(i \mid j)$ and $c(j \mid i)$ are intermediate results when calculating Q_{ij}. The procedure is shown in detail in Listing 3.

[4]Notice that to allow sliding windows to go from the very first to the very last event, additional padding needs to be applied on either side. We pad with the timestamps 0 and 9999999999, preventing any influence on time lags.

```python
def calculate_synchronization(location1, location2):

    # initialize the number of synchronous events for the two locations
    num_sync_events = 0

    # iterate over all timesteps in location1 using a sliding window
    for i_prev, i_current, i_next in sliding_window(location1, 3):

        # iterate over all timesteps in location2 using a sliding window
        for j_prev, j_current, j_next in sliding_window(location2, 3):

            # calculate the time delta between the current events
            current_diff = j_current - i_current

            # check if the current events occur simultaneously
            if current_diff == 0:
                num_sync_events += 0.5
                continue

            # calculate the time lag based on the two sliding windows
            time_lag = 0.5 * min(
                i_next - i_current, i_current - i_prev,
                j_next - j_current, j_current - j_prev)

            # decide whether the events are synchronous
            if 0 < current_diff <= time_lag:
                num_sync_events += 1.0

    return num_sync_events
```

Listing 1: Python pseudocode for a simplified event synchronization algorithm, applicable to any two series of events represented by epoch timestamps.

```python
def calculate_sync_strength(loc1_events, loc2_events):

    # calculate synchronized events between loc1 and loc2 (and reverse)
    loc1_sync = calculate_synchronization(loc1_events, loc2_events)
    loc2_sync = calculate_synchronization(loc2_events, loc1_events)

    # calculate the strength of synchronization
    sync_strength = (loc1_sync + loc2_sync) / \
        math.sqrt((len(loc1_events) - 2) * (len(loc2_events) - 2))

    # return the strength of synchronization
    # and the total and separate numbers of synchronous events
    return sync_strength, loc1_sync + loc2_sync, loc1_sync, loc2_sync
```

Listing 2: Python pseudocode for the calculation of the synchronization strength between any two series of events.

Improvements for the event synchronization algorithm

The implementation as shown so far naively processes all possible combinations of events when calculating the event synchronization measure. Given two event series i and j as well as their respective number of events s_i and s_j, the simple nested loop algorithm potentially calculates

	Latitude	13.375	16.375	34.375	34.375
Latitude	**Longitude**	**67.375**	**91.375**	**67.375**	**88.375**
13.375	67.375	1			
16.375	91.375		1		
34.375	67.375			1	
34.375	88.375				1

Table 3.4: "Empty" synchronization matrix for 4 exemplary locations (TRMM, 0.75°).

```
def calculate_sync_matrix(event_matrix):

    # calculate the sync strength for each permutation of grid cells
    for i in range(0, sync_matrix.shape[0]):

        # as the matrix is symmetrical, only calculate the upper half
        for j in range(0, i + 1):

            # calculate the synchronicity for the permutation of rows
            sync_strength, count = calculate_sync_strength(
                event_matrix[i], event_matrix[j])

            # save results in the respective matrices
            sync_matrix[i, j] = sync_strength
            sync_matrix[j, i] = sync_strength
            count_matrix[i, j] = count
            count_matrix[j, i] = count
            directed_matrix[i, j] = i_leads
            directed_matrix[j, i] = j_leads

    return sync_matrix, count_matrix, directed_matrix
```

Listing 3: Python pseudocode for processing an entire event matrix.

synchronicity up to $s_i * s_j$ times. This results in a lot of wasted computation because the actual portion of j that can be synchronous tends to be very small.

However, we can define a range of events in j for which we are sure that they cannot be synchronous and entirely skip their evaluation. According to our implementation, this holds for two cases: firstly, if the event in j occurs earlier than the event in i, it can be safely skipped, as the reversed iteration of the algorithm will deal with it. Secondly, an event in j can be skipped if the event in j occurs earlier than the partial time lag of the event in i, as the minimum of the full time lag calculation will never change. In such a case, we can additionally break the inner loop early, as further events in j would only occur even later. A version of the algorithm incorporating these improvements is shown in Listing 4.

Another (obvious) improvement would be to select only the few relevant events from j using advanced indexing and selection strategies, after which no more breaking or skipping in the loop would be needed. However, upon evaluation of such an approach, it became clear that this is not necessarily faster. Adequately selecting only relevant events from the series j necessitates some complex indexing and selection operations, which in turn depend on the existence of an

index or hash table for j. Creating such indices can take more than a second for a single series, making the approach especially slow for large matrices (as index creation is repeated i times).

```python
def calculate_synchronization(location1, location2):

    # initialize the number of synchronous events for the two locations
    num_sync_events = 0

    # iterate over all timesteps in location1 using a sliding window
    for i_prev, i_current, i_next in sliding_window(location1, 3):

        # calculate the last timestamp of location2 that could be synchronous
        latest = i_current + 0.5 * min(i_current - i_prev, i_next - i_current)

        # iterate over all timesteps in location2 using a sliding window
        for j_prev, j_current, j_next in sliding_window(location2, 3):

            # calculate the time delta between the current events
            current_diff = j_current - i_current

            # if the difference is negative, continue (too early)
            # the second pass will encompass these combinations
            if current_diff < 0:
                continue

            # check if the events occur simultaneously
            if current_diff == 0:
                num_sync_events += 0.5
                continue

            # break for timestamps that cannot possibly be synchronous
            # i.e. are much too late
            if j_current > latest:
                break

            # calculate the time lag based on the two sliding windows
            time_lag = 0.5 * min(
                i_next - i_current, i_current - i_prev,
                j_next - j_current, j_current - j_prev)

            # decide whether the events are synchronous
            if 0 < current_diff <= time_lag:
                num_sync_events += 1.0

    return num_sync_events
```

Listing 4: Python pseudocode for an improved version of the event synchronization algorithm, applicable to any two series of events.

A further performance consideration is the fact that the event synchronization calculation $c(i \mid j)$ is in itself an "elementary" operation that will provide results based on only two inputs i and j without any side effects. As a result, the entire synchronization matrix computation is heavily parallelizable, meaning that we can split the synchronization matrix into different jobs and run these jobs on different threads (or even machines). We have already implemented an (experimental) parallelized version of the event synchronization algorithm. The parallelized algorithm will, however, need further refinement to be fully applicable.

3.3.2 Building climate networks

Based on Section 3.2.2, the next step after the computation of a full event synchronization matrix is the transformation of said matrix into an adjacency matrix, allowing the creation of a network and its analysis using various methods and libraries. This transformation is a simple thresholding of the matrix using a specified percentile.

We further expand upon the work of Stolbova (2015) in two major ways: firstly, we build weighted networks in addition to the unweighted networks that have already been explored. To build a weighted network, values above the threshold are left as is, leading to links that are weighted with their corresponding synchronization measure. Contrarily, all values above the threshold are uniformly set to one when building an unweighted network. As a second addition, we build a directed network, meaning that we use an asymmetrical adjacency matrix based directly on the number of synchronous events (instead of combining both counts into a single synchronization coefficient). A directed network is much more appropriate for an analysis using PageRank, as the results of PageRank on an undirected network are relatively similar to the degree of such a network.

The creation of a climate network based on a synchronization matrix, as well as the calculation of the degree, betweenness, and PageRank for all nodes in the network, are a matter of a few lines of code as shown in Listing 5. Once the climate network computations have successfully finished, we are ready to visualize and interpret the results, which will be the major focus of the next section.

```
# extract the 95th percentile from a synchronization matrix
95th_percentile = np.nanpercentile(sync_matrix, 95)

# set all values above and below the threshold to 0 and 1
adjacency_matrix[adjacency_matrix > 95th_percentile] = 1
adjacency_matrix[adjacency_matrix <= 95th_percentile] = 0

# build a network from the resulting adjacency matrix
network = nx.from_numpy_matrix(adjacency_matrix)

# calculate the degree, betweenness and PageRank for all nodes
nodes_degree = network.degree()
nodes_betweenness = nx.betweenness_centrality(network)
nodes_pagerank = nx.pagerank_numpy(network)
```

Listing 5: Simplified Python pseudocode for the creation of a climate network from a synchronization matrix as well as the calculation of corresponding network measures.

3.4 Results & Evaluation

This section presents the results of applying what we have described in Section 3.3 to the TRMM dataset, more specifically the pre-monsoon, monsoon and post-monsoon seasons as extracted from TRMM. For each season, we show the results of degree, betweenness and PageRank measures in their weighted variation[5]. The presented PageRank results are based on the corresponding asymmetrical synchronization count matrices (directed networks). Knowing about the fundamental factors that influence the ISM (as introduced in Chapter 2), we give our thoughts about the patterns that can be seen in the results and how they could be connected to the fundamental behavior of the ISM.

3.4.1 Dataset

The results presented in this section are based on the TRMM dataset as introduced in Section 3.1. We have extracted data for the years 1998-2016, which corresponds to all years that were fully available at the time (March-December). The area included in the dataset is cropped to an extended overview over the Indian subcontinent (4.125-40.625N, 61.125-97.625E). Aggregation of the native 0.25° resolution to a lower 0.75° resolution yields grid borders equal to 4.375-40.375N, 61.375-97.375E (as shown in Fig. 3.1). The dataset was aggregated to reduce the computational effort required for event synchronization and climate network calculations. Going from 0.25° to 0.75° resolution effectively reduced the dimensions of the event synchronization matrices from 21609x21609 to around 2401x2401 (a reduction factor of 81x). The reduced resolution further ensures compatibility with the ERA-Interim dataset used later on.

3.4.2 Pre-monsoon season (MAM)

The results of applying our algorithms to the pre-monsoon season, namely the months of March, April and May (MAM), are shown in Fig. 3.5. We therein show the weighted network measures for a climate network thresholded by both its 95th and 90th percentile. The unweighted versions of all visualizations can be found in Appendix A.4.

The degree is strongest over the Indian ocean, enframing the Indian subcontinent from both east and west. On the western side lies the Arabian Sea, over which an area of high degree extends from the far west up to the natural barrier of the Western Ghats mountain range (WG). A similar but smaller area over the Bay of Bengal to the east of the Indian subcontinent ranges from the Indian coast onto the lands of Myanmar (BoB). Further patches of significant degree can be found over the Tibetan Plateau (TP), over north-eastern India and Nepal at the brim of the Himalayas (Himalayas) and areas of Pakistan and Afghanistan (NP, representing North-Pakistan).

Many patterns in the betweenness are much less clear and conclusive. There are, however, still spots that display significantly high betweenness, especially the area over Nepal (Himalayas). Additionally, smaller patches of high betweenness are sprinkled over the Tibetan Plateau (TP), to the north of Pakistan and over central India.

Our final measure, the PageRank algorithm, seems to follow the major patterns we have already seen in the degree. However, contrary to the degree, PageRank attributes very little

[5]The unweighted versions can be found in Appendix A.4. There is, however, only little difference between them.

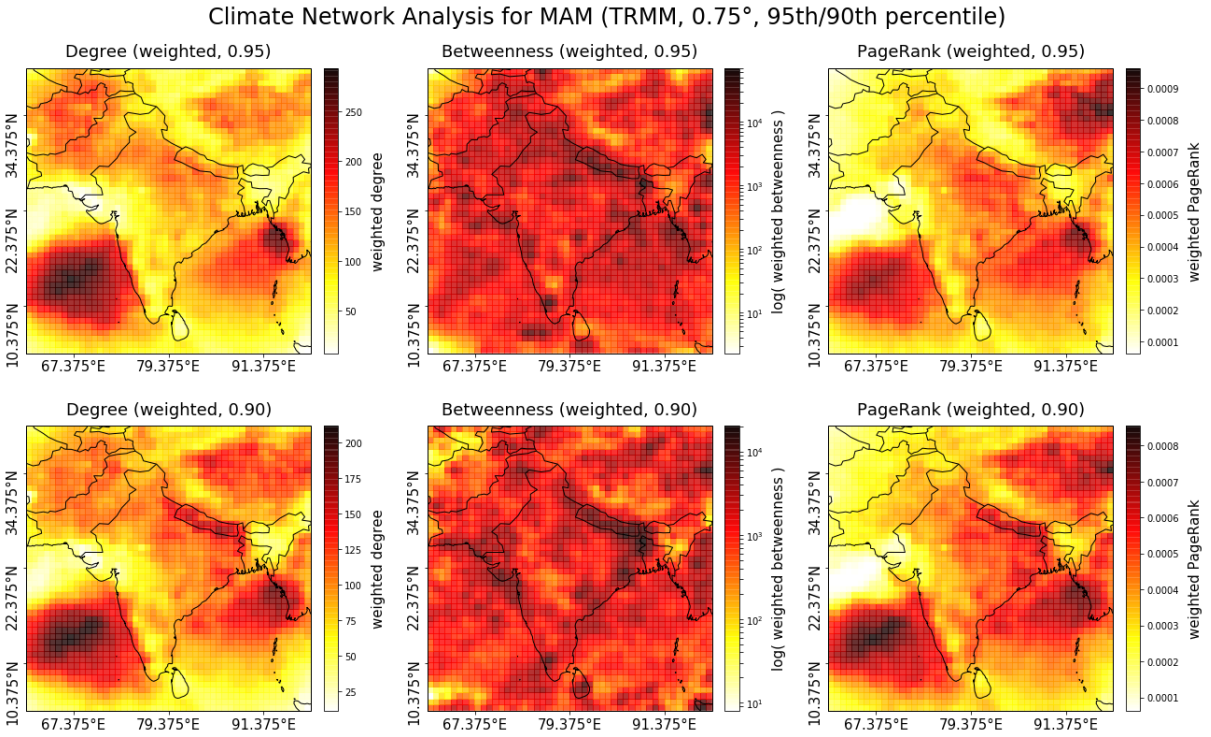

Figure 3.5: Weighted degree, betweenness and PageRank for the pre-monsoon season (MAM), thresholded with both the 95th and the 90th percentile. Based on the TRMM dataset at 0.75° resolution.

importance to the region to the north-west of India. Instead, central India and especially the Tibetan Plateau are ranked to be much more important.

During the pre-monsoon or "summer" season, rainfall in southern or central India is a rare occasion, as can be seen in Fig. A.8. Temperatures during these hot summer months range around 30-40°Celsius (Fig. A.6), even causing drought-like conditions in some years. This coincides well with our findings that the most central parts of the network are located in big clusters over the Indian Ocean, where most rainfall is concentrated during summer.

However, the centrality of a location does not (necessarily) represent the amount of rainfall but the overall "importance" of a location in the network, which is harder to reason about. For example, a location can obtain high degree centrality simply because it is part of many large-scale events like thunderstorms that cause high or even extreme precipitation in a large area, which would certainly explain the clusters over the ocean.

Furthermore, the significant degree and high betweenness at the brim of the Himalayas could be caused by its connectedness to either one or both clusters over the Indian ocean. As an area displaying high betweenness lies on a large portion of shortest paths in the network, a connection of the cluster over the Bay of Bengal to the Himalayas certainly makes sense. This would also be supported by the general importance of the Himalaya region as a "monsoon trough" that serves as an entry point to the subcontinent for moisture from the Bay of Bengal (see Chapter 2).

3.4.3 Monsoon season (JJAS)

Analyzing the months of the primary monsoon season (JJAS, June-September) as shown in Fig. 3.6 yields somewhat unexpected results: degree, betweenness, and PageRank all display a strong but compact pattern over Northern Pakistan (NP), especially when thresholded at the 90th percentile. Further areas of significant degree are located on the Tibetan Plateau (TP) and over the Indian Ocean around the southern tip of India (Arabian Sea/WG and BoB). High betweenness can be seen over Northern Pakistan (NP), near the Western Ghats (WG) and Kerala as well as on the other side of the southern tip of India.

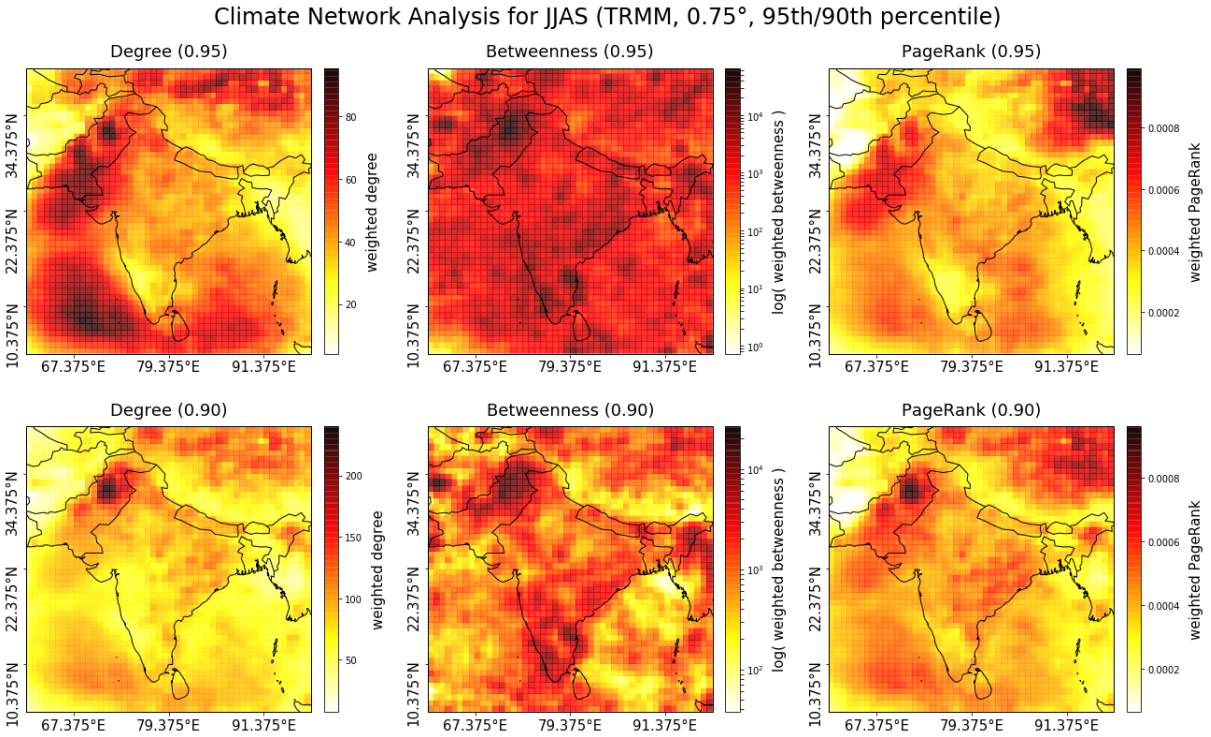

Figure 3.6: Weighted degree, betweenness and PageRank for the monsoon season (JJAS), thresholded with both the 95th and the 90th percentile. Based on the TRMM dataset at 0.75° resolution.

Having locations of high degree and high betweenness clustered over a region like Northern Pakistan suggests that even such a small region can be of great influence for many parts of the Indian subcontinent. However, it is hard to reason where the region's importance stems from. Perhaps it serves as a pivot between the Tibetan Plateau, the Arabian Sea and central India and thus is very central and part of many paths in the network.

The regions of strong betweenness flanking the Western Ghats could be explained by the vital role of the Kerala region in the overall onset of the monsoon season. Kerala is typically the first region of the Indian subcontinent that is reached by monsoonal rainfalls. Thus, it can be argued that extreme events in Kerala tend to lead events on the mainland and are most probably connected to many locations in India. Furthermore, the jet coming from the Arabian Sea is split into two streams due to the heights of the Western Ghats. Continuing into the mainland, these two streams could be a cause for the patterns of high betweenness on either side of the Western Ghats.

As it already has in the pre-monsoon season, the PageRank algorithm seems to attribute great importance to the Tibetan Plateau. The Tibetan Plateau is, in fact, one of the major drivers of the ISM, as its intense heating during the summer months and the resulting winds bring high amounts of moisture onto the subcontinent. With that being said, the plateau itself does not receive as much rainfall as many other parts of India, as we can see in Fig. A.8. The most probable reason for it being ranked as high is that it is strongly connected to the important region over North Pakistan.

3.4.4 Post-monsoon season (OND)

Contrary to the pre-monsoon season, the months of the post-monsoon season (OND, October-December) can be thought of as the Indian winter months. During these winter months, the ISM retreats completely, giving lead to the north-east monsoon towards the end of the year. The patterns as shown in Fig. 3.7 seem to largely track this retreat of the ISM.

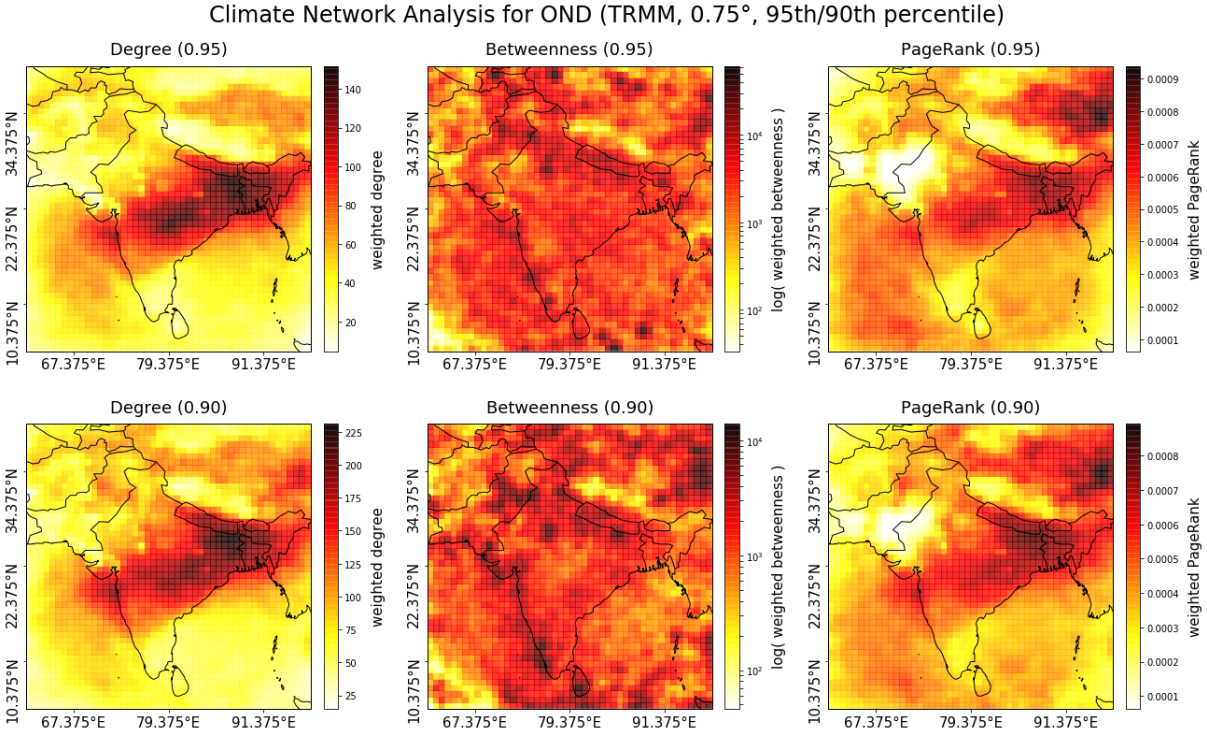

Figure 3.7: Weighted degree, betweenness and PageRank for the post-monsoon season (OND), thresholded with both the 95th and the 90th percentile. Based on the TRMM dataset at 0.75° resolution.

More specifically, a single area of very high degree encompasses entire central and north-eastern India, starting over the Arabian Sea (WG) and ranging well into Bangladesh. The degree in this area is at its highest over Nepal close to the Himalayas. The area further extends over the Tibetan Plateau (TP), although with less strength. Similarly, areas of high betweenness are located over Nepal, at the coast of the Arabian Sea, over North Pakistan (NP) and on the Tibetan Plateau (TP). The PageRank algorithm again seems to rank the Tibetan Plateau region as more important than the other two measures.

We mainly attribute the pattern of high degree and PageRank to the retreat of the ISM as well as the following buildup-phase of the north-east monsoon. During the later parts of the year, some locations experience more rainfall than even during monsoon. As the land cools down faster than the oceans surrounding it, the temperature gradient reverses, causing winds to change direction. This can dramatically change the spatial distribution of rainfall and could thus be a reason for the difference in patterns between the monsoon and post-monsoon periods.

3.4.5 Intercomparison with Stolbova (2015)

This section shows the results as they have been presented in the referred work (Stolbova, 2015) and compares them to the results we have obtained from our analysis. Next to indicating the validity of our approach, this might also show new properties that have only recently developed, as we have had access to four additional years of the TRMM dataset.

Differences in implementation

The calculations and results in Stolbova (2015) are based on unweighted climate networks with event series containing only the events above the 90th percentile. Furthermore, the climate networks are thresholded at the 95th percentile, leaving only the most statistically significant edges in a network. We have calculated the degree and betweenness measures in similar ways. There are, however, certain important differences and additions in our implementation that need to be taken into account when comparing the results of both works. These differences can be categorized into the following three categories:

PageRank and directed networks As a new measure, we have calculated the PageRank of climate networks, expecting to get further insights about the importance of locations. As the PageRank algorithm works best on directed networks, we have additionally computed directed variations of each network and network measure.

Weighted networks The climate network analyses as shown so far are based on weighted climate networks, where each link between two locations is weighted with the respective synchronization coefficient. While we have found this to represent certain details differently, the overall differences are small.

Dataset resolution We have aggregated the TRMM dataset to a 0.75° resolution by summarizing patches of 3x3 grid cells into one single grid cell. We have done this mainly due to performance reasons, as the full synchronization matrix for a native 0.25° resolution restricted to our area has dimensions of 21609x21609, making traversal of the matrix a computationally very complex effort (compared to a 2401x2401 matrix using 0.75° resolution).

Network thresholding As the much lower resolution of our TRMM input reduces the size of the networks by a large proportion, we have decided to perform an additional evaluation at the 90th percentile, trading off less significance for bigger networks.

Appendix A.4 contains an overview of the climate networks we have created, taking into account their differentiation into unweighted/weighted and directed/undirected networks.

Differences in results

While we have already shown the differences between the implementation of both Stolbova (2015) and our work, the network measures that finally result also exhibit some different characteristics. The visualizations as shown in Fig. 3.8 have been extracted from Stolbova (2015) and will now be used to provide an overview of these differences in our results.

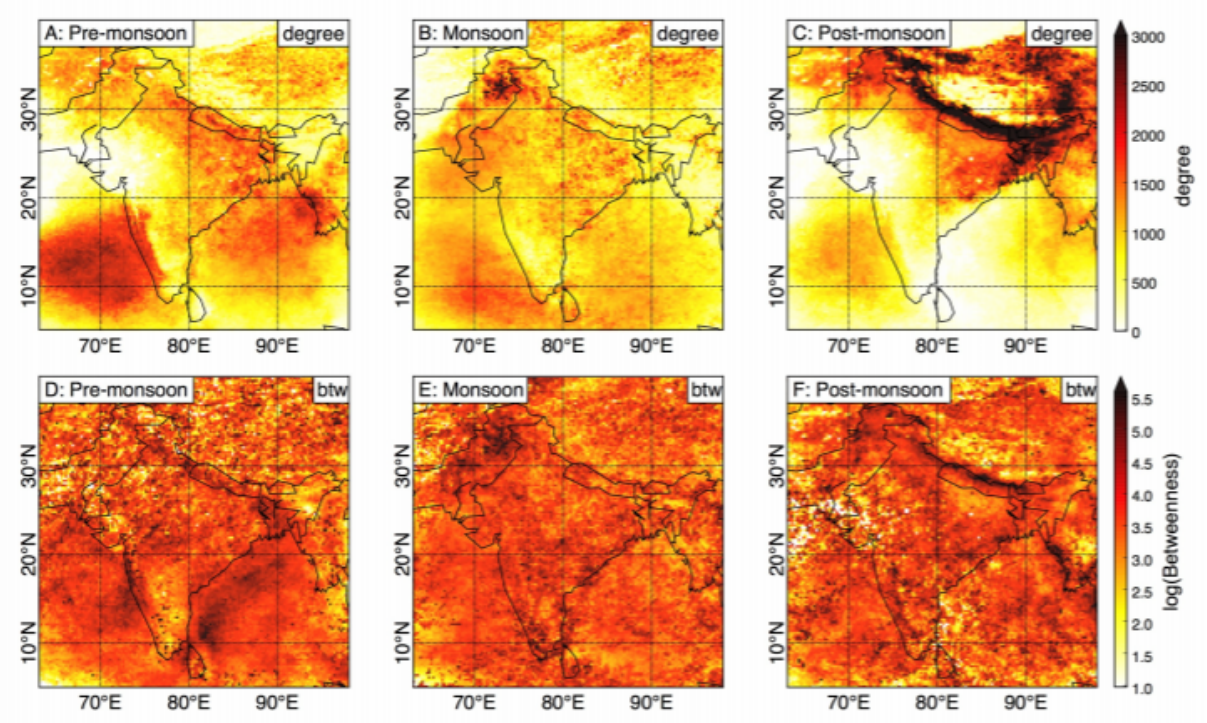

Figure 3.8: Degree and betweenness centrality as presented in Stolbova (2015). Based on the years 1998-2012 from the TRMM dataset at 0.25° resolution.

The visualizations as shown in Fig. 3.8 are obviously much more granular and detailed, as the full 0.25° resolution was used for the climate network computations, while we have used a 0.75° resolution. However, the general patterns seem to overlap well for the first two monsoon seasons. The main regions of interest during the pre-monsoon season are similarly located over the Arabian Sea, the Bay of Bengal, south of the Himalayas and on the Tibetan Plateau. The monsoon season shows the same strong clustering over North Pakistan and significant patterns at the southern tip of India as well as over the Arabian Sea.

Comparing the results for the post-monsoon season, we find that the discrepancies are much more significant. While the most central locations are clustered around the Himalayas in both Fig. 3.7 and Fig. 3.8, the area of high degree extends much farther to the west in our results. Additionally, the results as shown in Fig. 3.8 seem to be more concentrated to the Himalaya region and extend far up on the Tibetan Plateau. However, while the degree does not match all that well, the PageRank seems to accurately recognize the importance of the Tibetan Plateau during the post-monsoon season.

We mainly attribute the observed discrepancies to two important factors: the choice of resolution and the availability of data. We have used a lower data resolution to be able to speed up computation of the synchronization matrix and climate networks. It is obvious that a higher

resolution leads to much more detailed results. Furthermore, the choice of resolution probably also has an impact on the general structure of a network, as a 0.25° resolution results in a matrix that is much more sparse than it is after aggregation. An additional non-negligible factor is the availability of four additional years of TRMM data. We have been able to use the years 1998-2016 instead of 1998-2012, which corresponds to an increase of more than 25%. The trend that causes extreme events to occur more and more often would certainly also be capable of causing significant changes in the spatial distribution of extreme events, which might further explain the differences between our results.

3.5 Conclusion

In this part of our work, we have introduced the concepts of event synchronization and climate networks and have applied them for the analysis of extreme rainfall events on the Indian subcontinent. We made use of established network measures like betweenness centrality and PageRank to analyze the climate networks, allowing us to make observations about the importance of different regions on the Indian subcontinent (concerning extreme rainfall).

During the pre-monsoon season, the Indian Ocean and the Tibetan Plateau were identified as the most central regions in the network. We have attributed this to their importance during the reversal of the temperature gradient due to unequal solar heating and because most rainfall during the pre-monsoon season is concentrated over the oceans, while the subcontinent remains very dry.

For the monsoon season, a region over North Pakistan was surprisingly attributed the highest levels of centrality by all measures. While this is coherent with the findings in Stolbova (2015), it is nonetheless non-trivial to link to a specific climatic driver. We have suggested that the region might act as a pivot between several influential regions, leading to its high connectedness and centrality in the network.

The post-monsoon season has displayed the most significant differences from the results of Stolbova (2015): areas of high centrality and ranking span the entirety of central India and parts of the Tibetan Plateau, which we have linked to the retreat of the ISM as well as the starting transition to the north-east monsoon. The significant differences between the results of Stolbova (2015) and our own were attributed to the lower dataset resolution used in our work and, possibly, the additional TRMM years we were able to use.

When computing climate networks as shown in this chapter, the primary goal one has in mind is to analyze the spatial distribution of extreme rainfall and, based thereon, to potentially learn about patterns that are useful for a better understanding of the development of extreme rainfall events. Knowing how such events evolve and how an interconnection between regions on the Indian subcontinent could influence their development can be of great use in predictive efforts. Successful predictions of upcoming events could help to prepare for or even prevent the damage that is regularly caused by extreme rainfall events.

4

Part 2: Prediction of Monsoon Onset

The onset of the ISM plays a crucial role in both the daily lives of the Indian population as well as regarding the Indian economy overall. Whether the ISM is a blessing or a curse mostly depends on the timing, strength and durability of monsoonal rainfalls. While we have analyzed the distribution of extreme rainfall events in the previous chapter, this chapter is fully dedicated to the problem of ISM onset prediction. We focus on the monsoon onset over Kerala (MoK), as it marks the beginning of the yearly monsoon season for the entire Indian subcontinent.

Our approach to predicting the ISM onset is based on neural networks and their training on spatiotemporal meteorological datasets. An example of such a dataset is the TRMM dataset we have introduced and used in the previous chapter. As the TRMM dataset is focused solely on precipitation, we have found that it is not well suitable for our prediction task: many locations do not receive rainfall at any given time, making the data very sparse. As an alternative dataset, we introduce the ERA-Interim dataset in the first part of this chapter (Section 4.1). The ERA-Interim dataset provides many more features (e.g., temperature), many of which are less sparse, making them more suitable as training data for our models.

The neural network architecture we have found to work best is based on a new layer that has been introduced relatively recently. This new layer, the "ConvLSTM" layer, is based on a combination of convolutional and recurrent neural network concepts. We shortly introduce the main characteristics of ConvLSTM layers in Section 4.2.1. Following up in Section 4.3, we introduce the architecture of our final neural network model and present the experiments we have performed for its evaluation. We also provide an overview of the other model architectures we have tried but not developed further, as their accuracy was subpar.

4.1 The ERA-Interim Dataset

The ERA-Interim dataset is a global atmospheric reanalysis product created by the ECMWF. The center had already produced the popular ERA-40 reanalysis until 2002 and, during this process, had identified several issues in data assimilation that they addressed with the creation of the ERA-Interim reanalysis. ERA-Interim is also thought as a "bridge" between ERA-40 and a future reanalysis that spans the entire 20th-century (Dee et al., 2011).

Generally speaking, a reanalysis product is a dataset that provides many meteorological features through a single framework, for the same locations, and at the same resolution. Most often, reanalysis products are available on a global scale, which also holds true for ERA-Interim. The features provided by reanalysis datasets are not purely based on real observations, as this would often be unfeasible for the high spatial and temporal resolutions required. Instead, the

features are based on complex data assimilation algorithms and a combination of observations, simulations, and forecasts.

The ERA-Interim reanalysis provides hundreds of features for locations all around the world. ERA-Interim is available starting in 1979 and continues to be produced, albeit with a two-month delay. The native spatial resolution of the dataset is equal to approximately 0.75° on a geographical coordinate system.

To achieve consistency and comparability with the TRMM dataset, we subset the global ERA-Interim reanalysis into a gridded area between 4.5-40.5°N and 61.5-97.5°E. While this is slightly smaller compared to the area we have extracted from TRMM, the TRMM dataset shrinks to equal dimensions after the aggregation algorithm is applied (4.375-40.375°N, 61.375-97.375°E). After the described aggregation of the TRMM dataset, the two datasets are both in a 0.75° resolution. The final coordinate systems are offset by 0.125°, which is inherent to the datasets themselves.

While the ERA-Interim reanalysis provides a multitude of features, we focus on only a handful of features in this work. More specifically, we extract the temperature and relative humidity at a 1000hPa pressure level, the U and V components of wind at 700hPa and the U-component of wind at 200hPa. An overview of the extracted features is shown in Table 4.1, along with the corresponding identifiers we use later on. Appendix A.3.3 additionally shows the development of the most important features over the course of the pre-monsoon season.

ID	Description
msl	Mean-Sea Level Pressure
r	Relative Humidity at 1000hPa
t	Temperature at 1000hPa
u700	U-Component of Wind at 700hPa
v700	V-Component of Wind at 700hPa
u200	U-Component of Wind at 200hPa

Table 4.1: List of features from ERA-Interim and their identifiers as used in this work.

4.2 Related Work

Many of our experimental models are based on a relatively new approach to training neural networks on spatiotemporal data. We specifically refer to the work of Shi et al. (2015), where a newly defined *ConvLSTM* neural network layer was first applied to precipitation forecasting. Due to the reliance of many of our models on these layers, Section 4.2.1 is thought to provide intuition about the applicability of ConvLSTM networks. For the remainder of this work, we assume that the reader is familiar with neural network principles. However, as convolutional and recurrent layers are what ConvLSTM layers are based on, we explain some of their characteristics along the way.

4.2.1 Convolutional recurrent neural networks

ConvLSTM networks as introduced by Shi et al. (2015) were originally applied to the problem of precipitation nowcasting, the primary goal of which is described as giving a "precise and timely prediction of rainfall intensity in a local region over a relatively short period of time (e.g., 0-6 hours)" (Shi et al., 2015). In the aforementioned work, the authors try to predict a sequence of future precipitation radar maps from a sequence of past radar maps, essentially performing sequence-to-sequence prediction. These kinds of prediction tasks based on spatiotemporal data (where time, as well as geographical location, needs to be accounted for) are challenging, as a suitable neural network needs to be able to handle very high input and output dimensionalities.

An approach that is often used for sequence-to-sequence prediction is a two-layer architecture in which a first Long-Short Term Memory (LSTM) network reduces the input into a fixed-length sequence ("encodes" the input into an internal representation), while a second LSTM network then predicts an output sequence using as input the fixed-length sequence ("decoding"). This encoder-decoder framework was originally described in Sutskever, Vinyals, and Le (2014) and has been applied to many sequence-to-sequence tasks like machine translation. However, Shi et al. (2015) have found that the LSTM networks used for regular sequence-to-sequence prediction are not well suited to process spatiotemporal data. While they also use an encoding-decoding pattern in their models, Shi et al. (2015) propose a new type of network layer based on LSTM layers that can better take into account the spatial dimension of input values.

The newly proposed ConvLSTM network layer combines the concept of convolutions as applied in convolutional networks with the internal structure of regular LSTM layers. In a ConvLSTM network, the states in each LSTM-cell are represented by 3D-tensors with the same spatial dimensions as the input data. Updates to any hidden state are then performed by applying convolutions to both the input values and the previous state, meaning that the new state of any location is based on the previous and current values of its immediate neighbors (Shi et al., 2015).

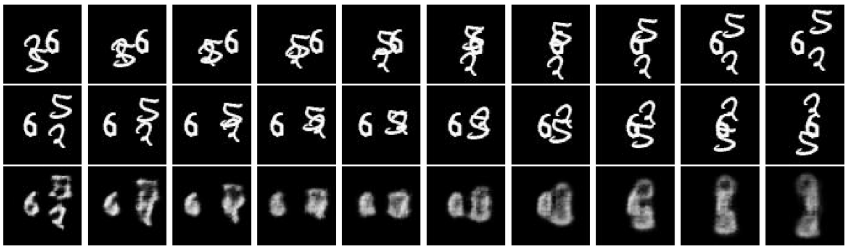

Figure 4.1: Results of predicting an "out-of-domain" moving-digits problem using ConvLSTM networks as depicted in Shi et al. (2015). From top to bottom: input frames, ground truth and predictions.

Based on the previously mentioned encoding-decoding pattern and their new ConvLSTM network layers, Shi et al. (2015) have built several models and evaluated them on both a generated dataset with moving digits (as shown in Fig. 4.1) as well as on a real radar map dataset. Over both of these evaluations, the ConvLSTM architecture outperformed a fully-connected regular LSTM architecture by a significant margin. Additionally, while the contours of the precipitation predictions were found to be blurred, they still significantly outperformed the benchmark concerning overall precision (Shi et al., 2015).

4.3 Prediction of Monsoon Onset using Neural Networks

Looking to find a model capable of learning the patterns leading up to the monsoon onset, as well as predicting the onset itself, we evaluated many different approaches to training neural networks based on spatiotemporal meteorological datasets like TRMM and ERA-Interim. Each approach had its advantages and disadvantages, many of which we only learned through trial and error and subsequently used to build improved architectures. Over many iterations with vastly different model architectures, we finally got to a model that seems to be able to learn patterns present before the monsoon onset and, based on these patterns, predict the monsoon onset with reasonable accuracy.

The first part of this section is thus focused on our certainly most important and useful result: our final working models and their evaluation. We describe their general architecture as well as the different evaluation schemes and parameters we have used. Furthermore, we evaluate how well the models have performed over the entirety of our experiments and assess the prediction capabilities from different points of view (e.g., how accurately can we predict on the 15th of May, or how accurately can we predict ten days before onset).

That being said, we would probably not have gotten to our final models without having tried many other architectures beforehand, iteratively improving upon the findings of each. The remainder of this chapter is then dedicated to a brief overview over all the model architectures we have evaluated during the creation of this work (in chronological order), along with a summary of our most important findings during the process.

4.3.1 E4: A working model based on ERA-Interim

The neural network architectures and hyperparameter tunings that we have found to work best during our experiments are based on two main ingredients: the ERA-Interim dataset with several of its features and multiple stacked convolutional recurrent layers (ConvLSTM2D). These models are all based on the latest Keras 2.0 and Tensorflow 1.4 Python libraries, which have greatly simplified our development, as even the relatively new ConvLSTM2D layer type is already implemented and usable from within the core libraries. The remainder of this section is dedicated to a more detailed description of the way we preprocessed the ERA-Interim dataset and built our final working model architecture. This specific model architecture is further also referred to as *E4*, the reasons of which will be explained later on.

Data preprocessing

One of the most important steps during the process of creating a neural network model (or any other machine learning model at that) is the preparation of the data that is to be used as its foundation. Without properly formatted input data, a neural network will typically not be able to learn any meaningful patterns. Over the course of our experiments, we tried different datasets (TRMM and ERA-Interim) and, more importantly, many heterogenous input formats, of which each had its advantages and disadvantages (we go over all of these evaluated input formats in their respective sections).

The approach we use in our final model architecture (E4) is most easily explained through an example: given time series data before the monsoon onset in an arbitrary year, we extract a fixed-length sequence (for example, a sequence of 60 consecutive days). The sequence ends at a given distance before the onset (e.g., 14 days before) and starts 60 more days before that. Feeding

this sequence (corresponding to a single training example) to our neural network model after training, we would like it to predict the number 14, which is comparable to a simple regression task. Intuitively, we ask our model the following question: "given data from the last 60 days, in how many days from today will the monsoon arrive?".

The actual preprocessing of the ERA-Interim dataset is slightly more complex: instead of a simple time series from which a sequence is extracted, we have a time series where each timestep is represented by a multidimensional matrix (a *3D-tensor* as shown in Fig. 4.2). For each location in the ERA-Interim coordinate grid as well as for each feature used from ERA-Interim (e.g., temperature), these tensors contain the value of the respective feature at the respective location (at that timestep).

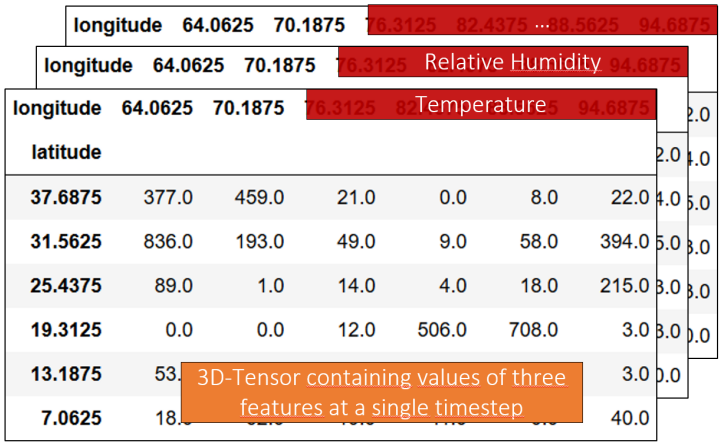

Figure 4.2: A single timestep as extracted from a training sequence. The 3D-tensor contains several features from ERA-Interim and, for these features, values at each location.

If we were to train our model only based on sequences that end 14 days before the onset, it would most certainly always predict the number 14, no matter the actual input. To train our model for different distances, we need to repeat the sequence extraction process many times per year, using a different distance to the monsoon onset in each repetition. Our most successful models are based on 30 training examples per year with distances in the range of $[1, 30]$ and a sequence length of around two months. In such a case, one of the training examples of each year ends one day before the respective onset ($t - 1$) and consists of the days until $t - 60$. A second training example ends at $t - 2$ and starts at $t - 61$, while a third training sequence ranges from $t - 3$ to $t - 62$. This is repeated up to a training example with data in the range of $t - 30$ to $t - 89$, yielding, in total, 30 training examples per year. This process is more intuitively shown in Fig. 4.3.

Normalization An additional step during data preprocessing is often the normalization of data into consistent ranges. Even though neural networks could learn without normalization, it often improves and speeds up their learning process. If different input features do not adhere to the same distribution (e.g., temperature and rainfall), the different magnitudes can lead to problems during the optimization process (when multiplying with the learning rate).

When processing the ERA-Interim dataset into a training, validation and test set, we apply statistical z-score standardization to center the mean of each feature at each location separately. To prevent the introduction of any bias, the mean and standard deviation of the training set are

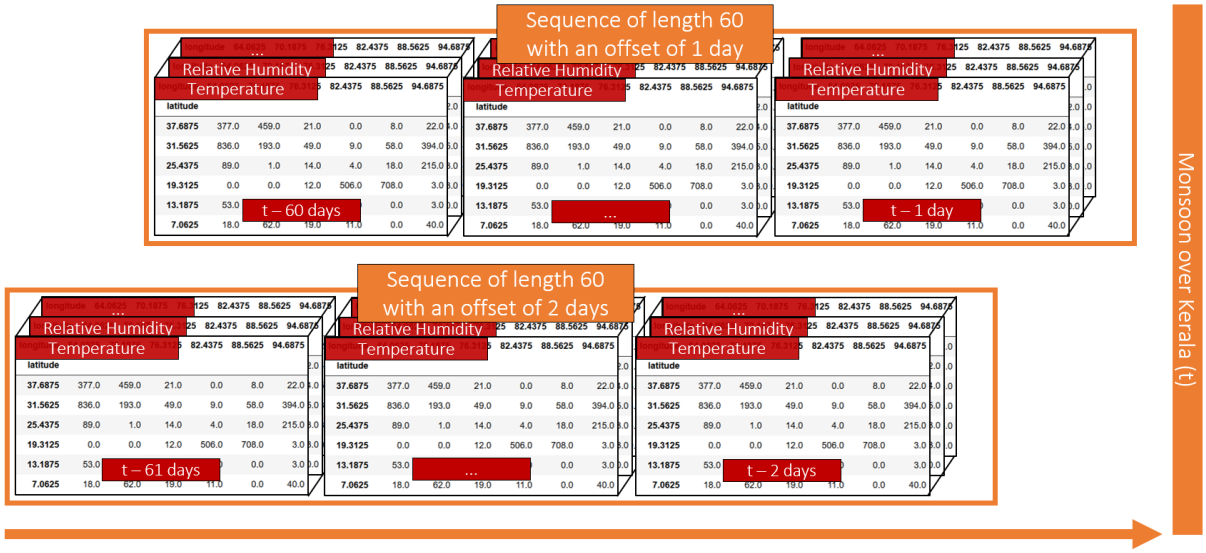

Figure 4.3: Two exemplary training examples based on three ERA-Interim features and a sequence length of 60 days. One training example consists of 60 timesteps, each of which is represented by a 3D-tensor as shown in Fig. 4.2.

reused when normalizing the values in the validation and test sets. This means that, to calculate the standardized value of feature f at location x/y in some timestep of a test sequence, the mean of said feature at said location over the training set would be subtracted, after which the result would be divided by the corresponding standard deviation.

Model architecture

It is not hard to see that a single ERA-Interim coordinate grid is inherently similar to a simple multichannel image: we have a height (latitudes), a width (longitudes) and multiple channels (e.g., temperature and relative humidity). When working with images, convolutional neural networks are by far the most prevalent approach. They have led to great progress in image recognition and other domains. However, 2D-convolutions are not directly applicable to multiple images ordered in a sequence (like a movie). Instead, they first need to be combined with recurrent network structures like LSTM.

We have already introduced a recent approach to the problem of learning from spatiotemporal data: the ConvLSTM2D neural network layer (Section 4.2.1). In our case, this layer performed much better than a simple stacking of convolutional and recurrent layers, making it the most important part of many of our models (including E4). With ConvLSTM2D at its core, the E4 model architecture learns by processing input sequences as shown in Fig. 4.3 using multiple ConvLSTM2D layers. From the sequence that these recurrent layers predict, the last state is flattened and passed through multiple fully-connected layers, the last one of which only consists of a single neuron. The network, therefore, predicts a single number, which intuitively corresponds to a regression model.

As an optional addition to the model, several time-distributed layers can be added in front of the core ConvLSTM2D layers. These time-distributed layers can apply any layer to a sequence and do so separately for each timestep in the sequence. For example, applying time-distributed Conv2D layers to a sequence of images would lead to a separate Conv2D layer with separate

weights for each image in said sequence. We can use these time-distributed layers to apply additional convolutional and pooling layers before feeding the sequences into the recurrent layers, adding further feature detection and dimensionality reduction capabilities.

An exemplary model based on the described architecture is shown in Fig. 4.4. Let us assume that the model therein has already been trained appropriately and is now used to perform a prediction on a single test example. The model passes this test example through two time-distributed layers consisting of Conv2D layers, each of which applies a 3x3 convolutional kernel and 16 filters (separately for each tensor in the input sequence). A further time-distributed layer applies a MaxPool2D layer, independently reducing each tensor in the input sequence to half its original spatial dimensions. The pooled sequences are passed through three ConvLSTM2D layers, each also with 16 filters and a 3x3 convolutional kernel applied to the state transitions. The last state of the final ConvLSTM2D layer (a single tensor) is flattened and passed through three 1024-neuron dense layers that are activated with the ReLU function, after which a linearly activated single-neuron layer outputs a numerical prediction.

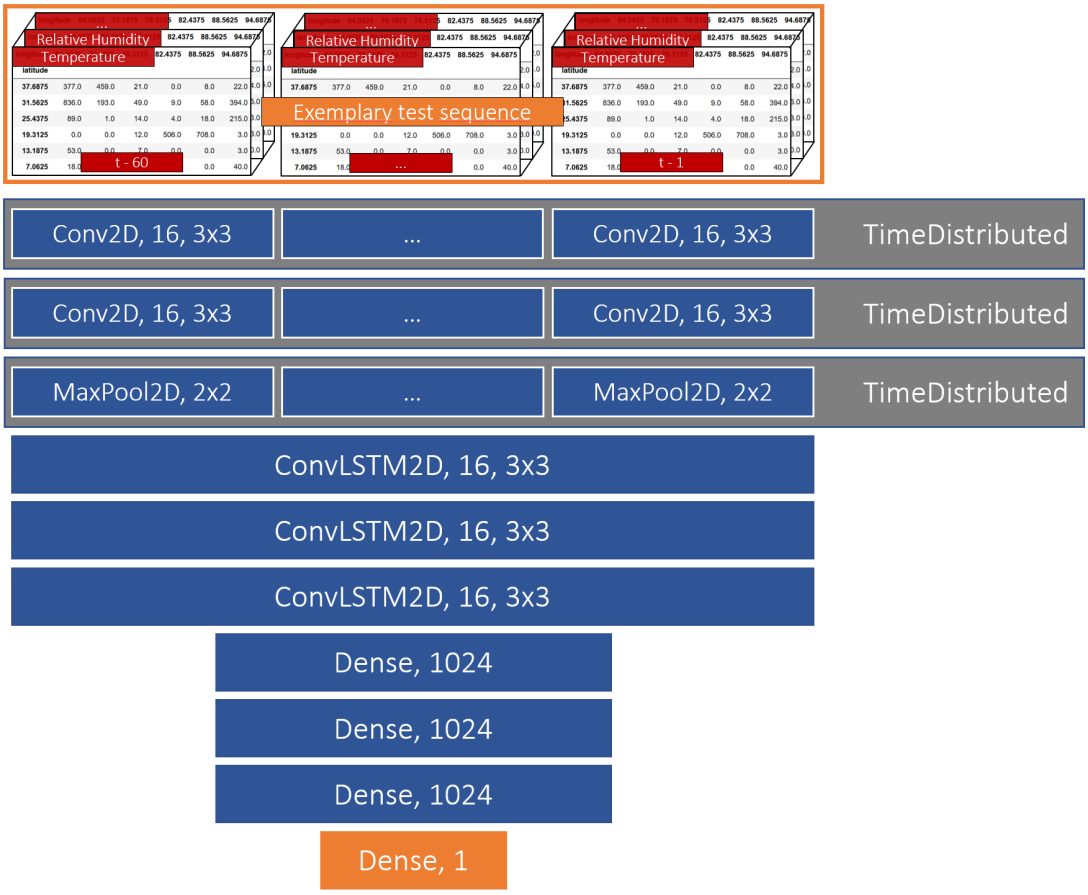

Figure 4.4: Exemplary model architecture of the E4 class (corresponding to the architecture used in EV07 and EV12 as introduced later on). An input sequence (e.g., a test example) with a structure as shown in Fig. 4.3 is passed through the different layers of the model and used to perform a numerical prediction using a linearly activated single-neuron output layer.

Regularization To reduce the tendency of the model to overfit to the training set, we apply L2-regularization to the kernels of all types of layers (Conv2D, ConvLSTM2D, Dense). This increases the loss whenever the model's internal representation of the problem gets too complex, effectively improving its ability to generalize. Also, we can and do most often use dropout after each layer. However, the exact regularization and dropout parameters vary with each tuning.

Batch normalization To speed up the training of our models, we make use of batch normalization after each layer in the models. During the training process, a batch normalization layer normalizes ("centers") the activations of each batch of training examples, preventing even small initial shifts in batch statistics ("internal covariate shift") to grow increasingly large when processed by many subsequent layers in a deep network.

Model evaluation

On our search for the best configuration to train our models with, we performed 45 experiments for this final model architecture alone. The parameter space we were searching through can be split into two major categories: "hyperparameters", referring to the parameters that are typically heavily tuned to improve model performance (e.g., learning rate, dropout rates, kernel sizes, etc.), and a second category that is more fundamental, as it already influences the data when it is being preprocessed. The choice of datasets and features thereof (see Section 4.1), as well as the amount of data that is being fed to the model (e.g., the sequence length), are exemplary parameters that we assign to the latter category. An overview of such "meta-hyperparameters" is shown in Table 4.2.

ERA-Features	msl/r/t	msl/r/t/u700/v700		msl/r/t/u700/v700/u200		
Onset Dates	IMD (1979-2017)			Objective (1979-2007) + IMD (2007-2017)		
Sequence Length	32	42	47	62	77	92
Sequence Offset	0-8	0-21	0-22	5-25	0-30	1-30

Table 4.2: "Meta-hyperparameters" as they have been used in the final E4-class models. IMD onset dates until 2007 are extracted from Singh and Ranade (2009) and concatenated with official IMD dates (2007-2017). Objective onset dates are extracted from the same source, but then also concatenated with IMD dates. The Sequence Offset specifies the range of offsets that is generated (meaning that for a value of 1-30, 30 examples are generated for each year with the smallest distance to the onset being 1 day). Details about the listed ERA-Features can be found in Table 4.1.

Training, validation and test set The ERA-Interim dataset was split into three fixed sets for each of our experiments: a training, validation and test set. As can be seen in Table 4.3, we originally started with only three years in the validation and five years in the test set (out of 38 years). When many models seemed to fit the training set very well, we decided to reduce the size of the training set, extending both the validation and test set (to four and eight years, respectively).

Because the "objective" onset dates are extracted from two different distributions (data that follows the objective definition of Singh and Ranade (2009) is available until 2007, after which the dates stem from the IMD), model performance was much worse when training on objective

onset dates using the second split. After it was reduced to the years 1979-2005, the training set did no longer contain any years based on the official IMD dates, while most of the validation and test set were based on these IMD dates. The entire training set was thus based on one distribution, while most of the validation and test years were based on another distribution. Unsurprisingly, the overall strongest overfitting was observed in the few experiments that were carried out this way.

As a remedy to the previously mentioned problem, we devised a third split scheme with a more appropriate distribution of years for experiments on both IMD and objective onset dates. This third split scheme was created by choosing a random year from each decade before the year 2000, and three years from each decade after the year 2000, without previous evaluation or analysis. The scheme was then successfully applied to the remainder of the experiments, allowing to train and compare models based on both IMD and objective onset dates.

	Split 1	**Split 2**	**Split 3**
Training	1979-2009	1979-2005	79/81-84/86-89/91-94/96-99/01-02/06-09/11-13
Validation	2010-2012	2006-2009	80/90/00/10/16
Test	2013-2017	2010-2017	85/95/03/04/05/14/15/17
Experiments	0-27	28-31	32-45

Table 4.3: The three different splits that have been applied to the dataset in the experiments for E4. Split 1 consists of 31 training years, 3 validation years and 5 test years. Split 2 consists of 27 training years, 4 validation years and 8 test years. The final split, Split 3, consists of 26 training years, 5 validation years and 8 test years, corresponding to 67%/13%/20% of the overall years.

Evaluation schemes Over the course of our experiments, we used various combinations of the previously described "meta-hyperparameters" (see Table 4.2), leading to what can be called *evaluation schemes*. An evaluation scheme is a group of experiments that are based on the same high-level parameters but have been tuned using different hyperparameters. We can reduce the total of our 45 experiments to 17 such evaluation schemes, each consisting of one to nine experiments. The listing in Table 4.4 shows these 17 evaluation schemes along with their "meta-hyperparameters". We regularly refer to these evaluation schemes in the remainder of this work.

Experimental conditions The experiments were run in different environments and are explicitly marked as such: either on a local GPU, the Kraken cluster or a Paperspace[1] virtual machine. Table 4.5 provides a more detailed overview of the environments used during the experimental phase. While all of the experiments for the E4-class of models were run on the *PS* environment due to performance constraints, most earlier experiments were run locally or on Kraken.

[1] https://www.paperspace.com/

ID	ERA-Features	Onset Dates	Seq. Length	Seq. Offset	#
EV01	msl/r/t	IMD	47	1-30	1
EV02	msl/r/t	IMD	62	1-30	2
EV03	msl/r/t	IMD	77	1-30	1
EV04	msl/r/t	Objective+IMD	62	5-25	3
EV05	msl/r/t	Objective+IMD	62	0-30	3
EV06	msl/r/t	Objective+IMD	62	1-30	3
EV07	msl/r/t/u700/v700	IMD	62	1-30	9
EV08	msl/r/t/u700/v700	Objective+IMD	32	0-30	1
EV09	msl/r/t/u700/v700	Objective+IMD	62	0-21	3
EV10	msl/r/t/u700/v700	Objective+IMD	62	0-22	8
EV11	msl/r/t/u700/v700	Objective+IMD	62	0-30	3
EV12	msl/r/t/u700/v700	Objective+IMD	62	1-30	2
EV13	msl/r/t/u700/v700	Objective+IMD	62	0-8	3
EV14	msl/r/t/u700/v700	Objective+IMD	92	0-30	1
EV15	msl/r/t/u700/v700/u200	Objective+IMD	42	0-30	1
EV16	msl/r/t/u700/v700/u200	Objective+IMD	62	0-30	1
				Total number of experiments	45

Table 4.4: Evaluation schemes for the E4-class models and the corresponding number of experiments, based on the "meta-hyperparameters" as shown in Table 4.2. Details about the ERA-features can be found in Table 4.1.

Environment	Trained on	Available memory
Kraken	CPU (24-cores)	128GB RAM
Local	GPU (GTX 970)	32GB RAM, 4GB GPU
Paperspace	GPU (Quadro P5000)	30GB RAM, 16GB GPU

Table 4.5: Runtime environments for the deep learning experiments.

Intermediary evaluation results

To provide a better overview of the outcome of our experiments, we reduce the results presented to only the best experiment of any given evaluation scheme (i.e., its best performing "hyperparameter tuning"). Table 4.6 shows the loss and validation loss after some key epochs as well as after the final epoch. Notice that loss and validation loss measures are based on the mean-squared error (MSE) but tend to be higher due to the regularization that is additionally applied.

Epoch	10		30		50		100		Final		
	Loss	Val. Loss	Loss	Val. Loss	Loss	Val. Loss	Loss	Val. Loss	Loss	Val. Loss	Epoch Nr.
EV01	48	132	37	85	30	87	15	72	11	68	500
EV02	43	92	33	70	25	73	13	57	6	63	500
EV03	54	76	35	63	24	87	14	96	6	69	500
EV04	43	73	32	40	24	36	13	56	6	17	500
EV05	69	86	41	54	25	45	17	26	12	24	500
EV06	41	59	32	58	24	58	14	35	5	33	500
EV07	72	125	52	92	39	66	18	52	10	38	500
EV08	82	225	50	137	27	112	13	73	7	66	500
EV09	41	38	41	39	40	37	39	38	38	37	227
EV10	63	76	37	38	20	46	9	16	7	20	150
EV11	61	65	37	63	23	30	19	49	19	26	300
EV12	56	105	41	99	27	80	18	44	10	49	500
EV13	6	10	7	5	7	5	6	5	6	5	117
EV14	199	246	95	141	103	212	87	77	90	898	134
EV15	196	263	92	137	38	77	23	58	14	61	300
EV16	124	139	84	268	53	194	31	147	9	144	300

Table 4.6: Loss and validation loss after a given number of epochs for the best experiment of each evaluation scheme. The best experiment is therein defined as whichever experiment has the lowest validation loss after its final epoch. While the majority of the experiments was run for 500 epochs, the worst-performing experiments were manually stopped earlier. It has to be taken into account that the range the offset can take on strongly influences the magnitude of the losses: in EV13, we had only 8 training examples per year, meaning that predictions should lie between 0 and 7 (a validation loss of 5 is thus very bad).

We can see that, for most of the evaluation schemes, the models were able to appropriately fit the training set. What greatly varies is the generalization capabilities of the different model architectures, which is typical when experimenting with neural networks. We would expect the final training losses to be even lower if regularization was omitted, which would, however, most certainly come at the cost of worse generalization.

During the 45 experiments of this stage, we have not applied any automated early stopping procedures. Such procedures are often used to interrupt the training process once the performance stops improving (or once the difference between loss and validation loss increases again). Some experiments have, however, been interrupted manually, once it became clear that their performance would most certainly only get worse (specifically EV09, EV13, and EV14).

Final evaluation

After the completion of the primary tuning phase and finding some model configurations that seemed to perform well, we wanted to reduce the variance in the test results. We thus performed a further round of evaluation where we retrained the best-performing models several times each and averaged their predictions. More specifically, we chose the five experiments that had the lowest validation loss (out of the experiments that used Split 3). These five experiments belonged to the EV02, EV06, EV07 and EV12 evaluation schemes, of which both experiments in EV02 were amongst the top five.

Evaluation procedure The final evaluations were performed as follows: first, the validation set was recombined with the training set, as it was no longer needed for tuning. This step allowed the final models to train on more data, potentially resulting in more representative results. The fixed validation set was instead replaced with a random 20% holdout set during training (changing every epoch). The five final model configurations were then trained five times each, using the same tunings and split schemes in each repetition. After the successful completion of all training rounds (consisting of 25 experiments), the predictions of repetitious experiments were averaged, yielding a final set of predictions. We hereafter refer to the final sets of experiments as 0-EV12, 1-EV07, 2-EV02, 3-EV02, and 4-EV06, corresponding to a unique identifier as well as the evaluation scheme they belong to.

Early stopping Contrary to the previous experiments, we applied early stopping with a patience of 130 epochs, meaning that if the validation loss did not decrease during more than 130 epochs, the training process was interrupted and the models immediately evaluated. Most experiments were stopped after around 200-300 epochs, while they could have run up to a full 500 epochs.

Test accuracy with respect to the distance from onset Looking at the averaged predictions of each set of experiments as shown in Fig. 4.5, we can see that the models trained on IMD onset dates are capable of predicting monsoon onset with quite a bit less than four days of mean-absolute error (MAE). Contrary to our previous beliefs that the objectively calculated onset dates would better represent the underlying patterns, the models based on these dates (0-EV12 and 4-EV06) seem to perform much worse: the accuracy when training on objective onset dates mostly ranges between 4-4.5 days MAE. This is partly caused by the fact that from 2007 onwards, the objective onset dates needed to be completed using official IMD dates. As

such, test years after 2007 are always evaluated on official IMD dates, while the model has mostly trained on objectively calculated dates.

A further interesting pattern that can be observed is that the models seem to predict more accurately (down to 2.5 days MAE) when predictions are made close to the monsoon onset, which makes intuitive sense. However, the predictions also seem to be more accurate in all models when predictions are made more than about three weeks before the onset. There might be predictive patterns present that tend to lead the monsoon onset by several weeks.

The first model from EV02 (2-EV02) seems to perform best overall, with a short-term accuracy of about 2.5 days MAE, a mid-term accuracy of 3-3.5 days MAE and a longer-term accuracy (up to one month before the onset) of around 2.5-3 days MAE. This is, by a fair margin, below the four days MAE that was described as the error of IMD predictions. Interestingly, this model is the only model that does not use additional convolutional layers and passes the inputs directly through ConvLSTM2D (after pooling), which might mean that these additional feature detectors are not as useful as they were thought to be.

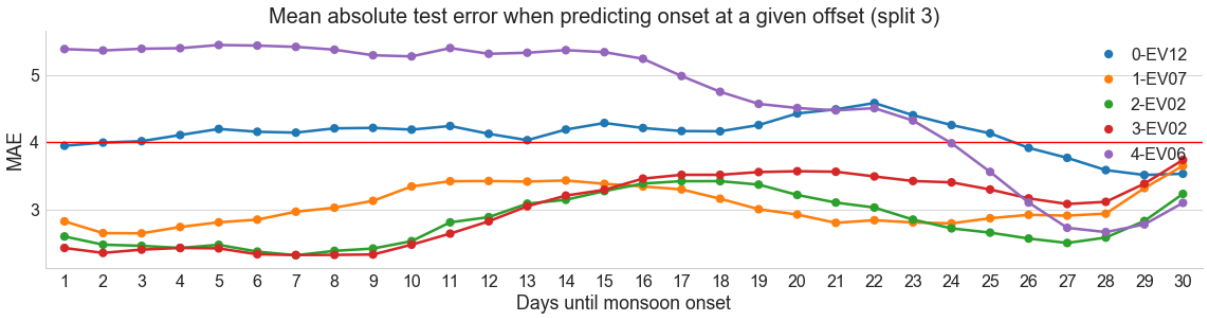

Figure 4.5: The prediction accuracy of our final sets of experiments at a distance of 1-30 days from the true onset for the years in the test set (1985, 1995, 2003, 2004, 2005, 2014, 2015, 2017).

Test accuracy for each test year A further interesting metric is the accuracy of the models when compared in-between years of the test set. Looking at the distributions in Fig. 4.6, we can see that there seem to be some years that are "easier to predict" than others. However, each set of models seems to display its best prediction skills in slightly different years. The year 2015 especially stands out, as the mean over all sets of models (even the two objectively trained ones, 0-EV12 and 4-EV06) is very similar. Further analysis of the accuracy on the test years can be found in Fig. A.17, where we show the accuracy of each set of models subject to a 95% confidence interval.

Test accuracy with respect to the prediction date The results as presented in Fig. 4.5 can be transformed in an interesting fashion: for each prediction made on the test set, we can assign the date the prediction would have been made according to the true onset date of the corresponding year. For example, given a prediction with a true distance of five days for the year 2017 (where the onset was on the 30th of May), we would end up with the 25th of May as a prediction date. Plotting these transformed results as shown in Fig. 4.7 allows us to see how accurate we can expect a prediction to be on any given day.

It is obviously much more useful to predict the onset earlier rather than later. According to Fig. 4.7, we might think that we would get our best predictions if we predicted towards the end

Figure 4.6: The prediction accuracy of our final sets of experiments for the years in the test set (1985, 1995, 2003, 2004, 2005, 2014, 2015, 2017).

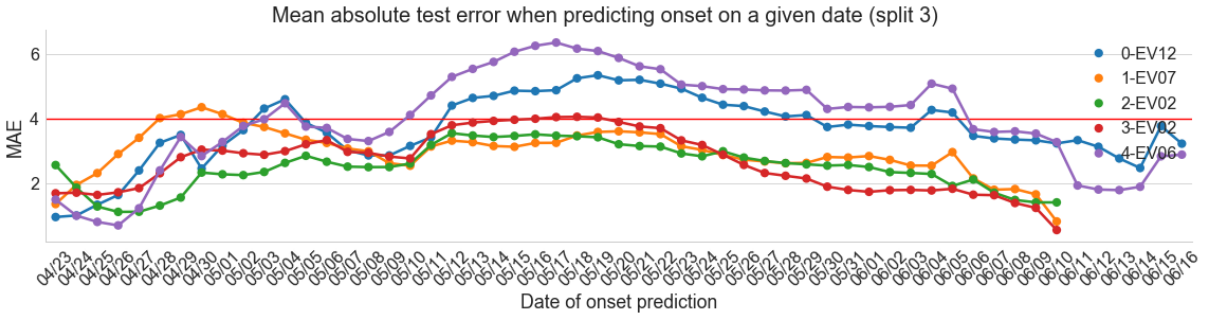

Figure 4.7: The prediction accuracy of our final sets of experiments at any given date before the true onset for the years in the test set (1985, 1995, 2003, 2004, 2005, 2014, 2015, 2017).

of April. However, it has to be taken into account that not all dates in this visualization consist of the same number of predictions. The real onset dates range from dates in the second half of May to dates in mid-June, meaning that the 23d of April will only rarely be included in training or test examples. The same holds for prediction dates in early June, as they would lie after the real onset in some years. As such, predictions performed around or after the 8th of May would most certainly lead to more accurate results than predictions in April, as most training examples would include these dates. The prediction task could also be repeated every day after that, as our model is in its capabilities not restricted to only short- or long-term predictions.

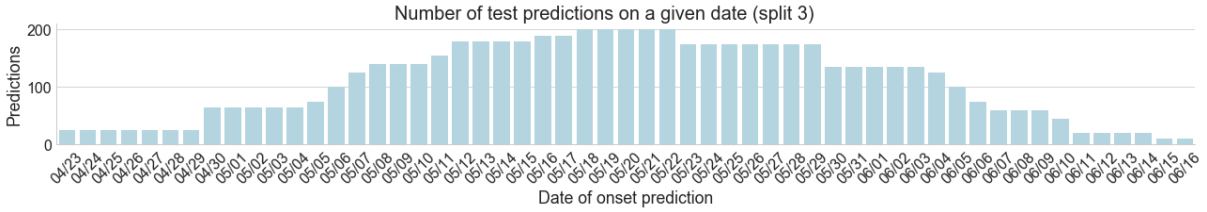

Figure 4.8: Number of predictions for any given prediction date for the years in the test set (1985, 1995, 2003, 2004, 2005, 2014, 2015, 2017).

Overall accuracy When we combine the separate measures shown previously, we get a single accuracy measure over the entire test set. As shown in Table 4.7, 2-EV02 is the overall most accurate model, achieving an accuracy of around 2.8 days MAE (with 2.46 days MAE as the

minimum and 3.09 as the maximum over all five repetitions). The other two models that are also based on IMD onset dates (1-EV07 and 3-EV02) follow the first model closely, while the models based on objective onset dates (0-EV12 and 4-EV06) are significantly lagging behind. However, this might look different if objective onset dates were available for the entirety of the training and test years (i.e., 1979-2017).

Final evaluation set	**0**	**1**	**2**	**3**	**4**
Evaluation scheme	EV12	EV07	EV02	EV02	EV06
Onset dates	Objective	IMD	IMD	IMD	Objective
Maximum mean-absolute error	4.49	3.50	3.09	3.29	5.69
Averaged mean-absolute error	4.11	3.05	2.82	3.03	4.25
Minimum mean-absolute error	4.00	2.60	2.46	2.63	3.87

Table 4.7: Overall minimum, averaged and maximum mean-absolute error for the final models.

4.3.2 T1-T5 and E1-E3: Other model architectures

On the path towards the well-working E4-class of models, there were many other approaches that disappointed with their learning capabilities. However, even though they did not end up to be a working model, each model architecture we have evaluated and subsequently discarded has led to important findings, without which we could not have created a working model at all. In this section, we walk through these different variations of neural network models and shortly summarize the key takeaways of each variation.

The model variations we have evaluated differ in their network structure, in the data and features used to train the models or simply in the tuning of hyperparameters applied. Table 4.8 summarizes the key characteristics of the models we have built and evaluated and shows how the E4-class has come to be. Each model architecture is assigned a unique version identifier that we use when referring to the specific architecture (like we already did with *E4*). The identifiers are based on the type of dataset used to train the model and the variation of the model architecture used (e.g., *T1*: TRMM v1, *E3*: ERA-Interim v3). All of the models presented were built and trained using the latest Keras 2.0 and Tensorflow 1.3/1.4 libraries for Python.

T2: Naive classification models using TRMM

The first models we evaluated were classification networks based on the TRMM dataset (see Section 3.1) at various spatial resolutions (ranging from a 0.25°x 0.25°resolution to an aggregated 2.5°x 2.5°resolution).

Data preprocessing A single TRMM cell (i.e., a single location on the grid) was taken at a time and transformed into a time series for that specific location (from the first of May up to the prediction date). The latitude and longitude indices were appended in front of the time series to allow the network to learn the spatial features of each example. This led to training examples each consisting of a one-dimensional fixed-length sequence containing coordinates and daily

Model	Architectural Key Points
T1	First trials with LSTM and TRMM; No experiments; Discarded
T2	Each location of each year as a separate example; Sequences of values for single grid cells processed with LSTM; Categorical prediction on the 11th of May (predicted MoK in 1-40 days)
T3	Each year as a separate example; One sequence of TRMM grids (daily values) per year processed with ConvLSTM2D; Numerical prediction on the 11th of May (predicted MoK in x days)
T4	Regularization for dense, convolutional and recurrent layers; Based on T3
T5	Objective or IMD onset dates from Singh and Ranade (2009); Prediction on the 22nd of May; Based on E2
E1	Training on the ERA-Interim dataset and multiple features (temperature, relative humidity); Based on T4
E2	Objective or IMD onset dates from Singh and Ranade (2009); Prediction on the 22nd of May; Based on E1
E3	Improved structuring of Python classes; Extraction of core model logic into base classes; Same functionality as E2; No experiments
E4	Multiple examples per year with variable offset to the onset date; Choice of additional input features; Possibility for additional 2D-convolutions before ConvLSTM2D; Switch to the functional Keras API; Based on E3

Table 4.8: Overview of all neural network models that have been built & evaluated

rainfall amounts as well as a label corresponding to its category. Table 4.9 provides an overview of the structure of the examples used to train and evaluate the model.

The labels were represented as one-hot encoded vectors like $[0, ..., 1, ..., 0]$. They were calculated based on the difference in days between the fixed prediction date and the monsoon onset date over Kerala. As the earliest onset in the used source (Ordoñez et al., 2016) for the relevant period (1998-2017) was found to be on the 12th of May, the prediction date was set to be the 11th of May. With the latest historical onset during the available period (1887-2017) being on the 15th of June, the category labels were naturally restricted to a range of $[1, 35]$. Some additional padding was added to allow for future late onsets, yielding a range of $[1, 40]$. Due to the focus on the onset date over Kerala, the examples for each year could all be assigned the label for the respective year.

Modeling To build a model based on the preprocessed examples, a sequence of LSTM and Dense layers was trained on the training set. A final softmax layer allowed the prediction of classes. Losses were thereby calculated using categorical cross-entropy. A percentage of training data was randomly held out in each epoch to validate the intermediary results. Listing 6 shows a simplified exemplary model architecture based on the default parameters.

Latitude	Longitude	01.03.1998	02.03.1998	...	11.05.1998	Label
6.375	63.625	0.27	0.09	...	3397.29	[0, ..., 1, ..., 0]
6.375	66.125	0.03	10.08	...	4632.24	[0, ..., 1, ..., 0]
...	...	...	...	...	...	[0, ..., 1, ..., 0]
36.375	91.125	0.79	0.00	...	68.61	[0, ..., 1, ..., 0]
36.375	93.625	11.44	0.06	...	40.06	[0, ..., 1, ..., 0]

Table 4.9: Exemplary list of training examples for the T2 architecture for the year 1998 at 2.5°aggregation.

```
# build a new sequential model
model = Sequential()

# add an LSTM layer for initial input transformation
model.add(LSTM(128, return_sequences=False))

# add several dense layers
# optionally add dropout after each layer
model.add(Dense(256, activation='relu'))
model.add(Dense(512, activation='relu'))
model.add(Dense(256, activation='relu'))

# add a softmax layer for classification
model.add(Dense(40, activation='softmax'))

# compile the model
model.compile(
  loss='categorical_crossentropy',
  optimizer='rmsprop',
  metrics=[top_k_categorical_accuracy])
```

Listing 6: Simplified Python pseudocode for an exemplary T2 model (default configuration).

Preliminary results The presented classification approach was soon discarded due to poor overall learning capabilities. Firstly, classification was found not to be a suitable approach for prediction of an upcoming onset date. An adequate loss should incorporate the numerical distance in days between the predicted and the true onset date, which is not easily possible when evaluating classification using cross-entropy loss. The implementation of a custom loss function could remediate some of these concerns but was not regarded as useful at the time.

Secondly, using each grid cell as a separate example might have led to a larger number of training examples overall, but rainfall in separate single geographical locations in India probably cannot be seen as a reasonable predictor for a large-scale phenomenon like monsoon onset. It cannot be ensured that the model would be able to learn any spatial relationships based on the inclusion of coordinates as the very first timesteps in each sequence. As we have already explored in Section 2.2, the behavior of Indian Summer Monsoon is influenced by many external factors that are not necessarily restricted to the Indian subcontinent. Thus it is most probably a combination of many places (some of which more important than others) that could lead to successful onset date predictions.

A further disqualifying factor against the use of any such classification approach was the fact that the model would be unable to express any future onset occurring outside the borders of the classification (e.g., more than 40 days after the prediction date). As we have elaborated in Section 2.3, the behavior of the ISM is ever-changing, and it cannot be ruled out that the onset date distribution will change significantly in the time to come.

T3 & T4: First convolutional models using TRMM

The findings of the first naive classification suggested that the overall model architecture and the data input shape needed to be rethought. To be able to capture temporal as well as spatial relationships between different geographical locations in India, the input time series needed to be changed such that each timestep would contain daily data for the entire Indian subcontinent.

Data preprocessing To extend the dimensionality of the input data, the TRMM dataset was transformed to a fixed-length sequence of coordinate grids (each coordinate grid thereof containing data for all locations). This resulted in only a single training example per year of data. As TRMM data is available from 1998 onwards and with some data that would be held out for validation and testing, the number of training examples shrunk to roughly a dozen.

Furthermore, keeping the outcome calculations the same as for the previous classification task, the classification was changed to numerical regression. Instead of trying to predict one of the classes in the range $[1, 40]$, the model would thus predict an exact number that could be both higher or lower than numbers in the classification range. However, the prediction would still represent the predicted number of days from the prediction date until the occurrence of monsoon onset over Kerala.

Lat./Lon.	63.375	64.375	...	95.375	96.375
6.125	849.779974	791.189987	...	139.547018	430.566095
7.125	323.249996	393.899986	...	38.789999	172.289993
...	...	...	...	...	...
38.125	0.000000	0.000000	...	1.648755	7.281448
39.125	0.000000	0.000000	...	1.584739	1.154755

Table 4.10: An exemplary coordinate grid aggregated to 1.0°. A sequence of 72 such matrices and its label correspond to a full training example for the T3/T4 models (one year of data).

Modeling To be able to process a sequence of grids (i.e., matrices), the first layers of the model needed to be able to handle multi-dimensional input sequences. Said layers were therefore changed from a regular LSTM layer to Convolutional LSTM layers (see Section 4.2.1 on ConvLSTM2D layers). These layers can process a sequence of multi-dimensional input matrices and learn complex spatial as well as temporal relationships.

The new model architecture (T3) further included batch normalization to improve training speed. Furthermore, in comparison to the previously used cross-entropy loss, mean-squared or mean-absolute error could now provide a reasonable distance metric. Listing 7 shows an exemplary model configuration using default parameters.

A later extension (T4) added regularization for the convolutional, recurrent and dense layers to improve the models' ability to generalize. Furthermore, the validation set was changed from a random percentage of the training set to a fixed set of years. The validation years were defined to be the latest in the training set such that long-term trends could be captured by the model, which might not have been possible using the previous random validation split.

```
# start building a sequential model
model = Sequential()

# add a first convolutional LSTM layer
model.add(ConvLSTM2D(filters=32, kernel_size=(7, 7), activation='relu'))
model.add(BatchNormalization())
model.add(Dropout(0.4))

# add a second convolutional LSTM layer
model.add(ConvLSTM2D(filters=16, kernel_size=(5, 5), activation='relu'))
model.add(BatchNormalization())
model.add(Dropout(0.4))

# add a third convolutional LSTM layer
model.add(ConvLSTM2D(filters=8, kernel_size=(3, 3), activation='relu'))
model.add(BatchNormalization())
model.add(Dropout(0.4))

# add max pooling before flattening to reduce the dimensionality
model.add(MaxPooling2D(pool_size=(4, 4))

# flatten to make data digestible for dense layers
model.add(Flatten())
model.add(BatchNormalization())

# first dense layer
model.add(Dense(1024, activation='relu'))
model.add(BatchNormalization())
model.add(Dropout(0.6))

# second dense layer
model.add(Dense(512, activation='relu'))
model.add(BatchNormalization())
model.add(Dropout(0.6))

# third dense layer
model.add(Dense(256, activation='relu'))
model.add(BatchNormalization())
model.add(Dropout(0.6))

# single linear neuron for numerical prediction
model.add(Dense(1))

# compile the model
model.compile(
  loss='mean_squared_error',
  optimizer='rmsprop',
  metrics=['mean_squared_error', 'mean_absolute_error'])
```

Listing 7: Simplified Python pseudocode for an exemplary T3 model (default configuration).

Preliminary results Many experiments with various hyperparameter tunings showed that the first variation of the model architecture (T3) was unable to fully learn the concepts of the training set and that it did not generalize to the validation set very well.

The results of experiments for the T4 variation were much improved, especially if regularization was used for all types of layers (in addition to dropout). In training, the model converged up to a certain point but tended to plateau afterward, even if run for hundreds of epochs. This would normally suggest that the optimization process has reached either a minimum or is otherwise unable to learn any further, except that in this case, the validation loss plateaued as well (instead of increasing due to a growing tendency to overfit).

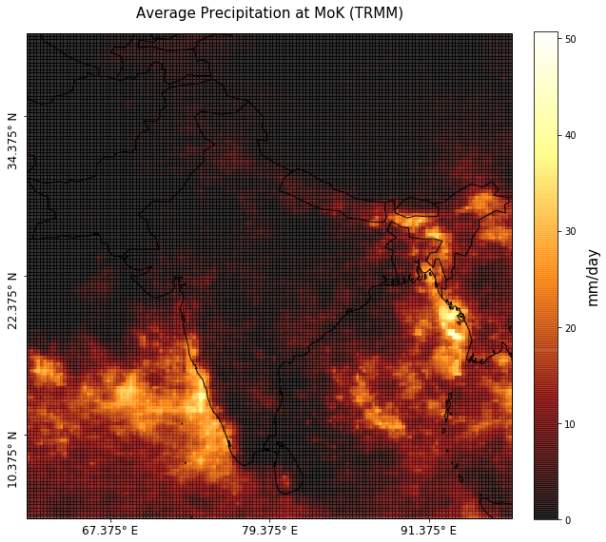

Figure 4.9: Average precipitation at monsoon onset over Kerala (TRMM, 1998-2017)

We found that, in this case, the models learned a way to get close to the real outcomes without learning anything particularly useful about the patterns in the input data: all the outcomes of the validation and test sets were simply predicted to be the mean value of the outcomes in the training set. There could be several explanations for this behavior, of which most suggest that something is off with the input data. It might be that the sparsity of the TRMM dataset (as visualized in Fig. 4.9) and its many places without any rainfall prevent the neural network from learning any useful patterns. Alternatively, it could be that we are trying to find patterns in the input data that don't exist in the way we would have imagined (i.e., that we are trying to fit a model to noise).

E1: Convolutional models using ERA-Interim

The overall results from T1 up to T4 suggested that the TRMM dataset on its own might not be well suited for the prediction task at hand. An obvious way of improvement was to use another dataset for the prediction tasks, namely the ERA-Interim reanalysis dataset.

Compared to the TRMM dataset, the ERA-Interim dataset brings with it several advantages relevant to our task. Firstly, much more data is available (dating back to 1979 instead of 1998), which can improve the performance of deep neural networks significantly (even though the number of total examples is still comparatively small). Secondly, there are many more features

(of which some could prove to be better predictors than solely the amount of daily rainfall). Furthermore, the dataset had already been used for onset prediction tasks in (Stolbova, 2015).

A disadvantage of using the ERA-Interim dataset might be that its spatial resolution is much lower than the one of the TRMM dataset (0.75°x 0.75°compared to 0.25°x 0.25°). When training with the ERA-Interim dataset, the input matrices in the sequence would thus be much smaller in the spatial dimensions (latitude and longitude), providing the models with less detailed data to train on. However, many more ERA-Interim features could be added as channels to the model inputs, giving the models the possibility to learn more complex patterns and relationships. The lower resolution of the ERA-Interim dataset further means that each input matrix is up to 9 times smaller than with TRMM, leading to lower overall computational effort and hardware requirements (i.e., memory capacity).

Data preprocessing This first iteration of the ERA-based model (E1) made use of only two features from ERA-Interim: temperature and relative humidity each at a 1000hPa pressure level. Stolbova (2015) found this combination of features to be a reliable predictor of the monsoon onset date. While this finding should also apply to our problem, the approach used was based on statistical time-series analysis and prediction and as such might not directly transfer to the training of our neural network models.

To feed temperature and relative humidity to the model simultaneously, the features needed to be stacked in a single coordinate grid. Each grid cell would thus contain multiple channels (i.e., features, same as the RGB channels in images) and be represented as a 3D-tensor (i.e., a "3D matrix"). Combining grids to a sequence yielded a sequence of 3D-tensors (a 4D-tensor) that could then be used as input for the neural network models.

Modeling Incorporating these findings into a new model structure, we extended the convolutional and regularized T4 model to be able to learn from a sequence of 3D-tensors. The convolutional layers would then process these inputs just like they would process an RGB image with its three channels. The architecture for this is shown in Listing 8.

Preliminary results Running experiments for the E1 model yielded results that displayed similar patterns to the TRMM-based models evaluated before: the models were learning up to a certain point and plateaued at unsatisfactory levels of loss and validation loss. They still seemed to get stuck during the learning process.

As experiments for entirely different datasets (TRMM and ERA) and many hyperparameter tunings for each showed similarly unsatisfactory patterns, it was probable that something was off with the data being fed into the model. Further research showed that the distribution of onset dates used seemed to show some irregularities that had not been regarded as irregular before: many onset dates were set in the first half of May (as early as on the 12th of May), which is highly improbable for typical ISM onsets[2]. These onset dates might, for example, have been caused by bogus monsoon onsets that were not registered as such at the time.

The labels of the training, validation and test examples were all calculated using the skewed onset dataset. The early outliers in the dataset forced us to predict even earlier, which is why predictions up to this point were always made on the 11th of May, and why the sequences were restricted to the 72 days in between 01.03-11.05 of each year.

[2]This was brought to light during consultation with V. Stolbova.

```
# start building a sequential model
model = Sequential()

# add a ConvLSTM2D layer
model.add(
  ConvLSTM2D(
    filters=16,
    kernel_size=(7, 7),
    activation='tanh',
    recurrent_activation='hard_sigmoid',
    kernel_regularizer=regularizers.l2(0.02),
    recurrent_regularizer=regularizers.l2(0.02)))

model.add(MaxPooling3D(pool_size=(1, 2, 2)))
model.add(BatchNormalization())
model.add(Dropout(0.6))

# ... repeat with 5x5/8 and 3x3/4

# flatten to make data digestible for dense layers
model.add(Flatten())
model.add(BatchNormalization())

# add a new dense layer
model.add(Dense(1024, activation='relu', kernel_regularizer=regularizers.l2(0.02)))
model.add(BatchNormalization())
model.add(Dropout(0.6))

# ... repeat with 512 and 256 nodes

# final dense layer for numerical prediction
model.add(Dense(1))

# compile the model
model.compile(
  loss='mean_squared_error',
  optimizer=RMSprop(lr=0.01),
  metrics=['mean_squared_error', 'mean_absolute_error'])
```

Listing 8: Simplified Python pseudocode for an exemplary E1 model (default configuration).

E2, E3 & T5: Convolutional models with objective onset dates

After further analysis of onset distributions for different onset datasets, the objective definition of monsoon onset proposed by Singh and Ranade (2009) seemed to be the most adequate for our training at the time. The distribution seemed to be much less skewed overall, with more similarities to a uniform distribution than to a Gaussian normal distribution, at least when data was restricted to the relevant area between 1979-2017 (see Appendix A.2 for the full analysis). Notice that we found out only during the final evaluation of the E4 models that the IMD onsets calculated by Singh and Ranade (2009) work even better, especially as they need to be combined with official IMD dates after 2007 (yielding one "single" distribution).

While it might be counterintuitive that the ISM onset would not follow a normal distribution (as many natural processes do), a more balanced distribution of labels can improve the performance of machine learning models. Many models work better if input data is preprocessed with stratification methods (i.e., classes are rebalanced), as the model might otherwise tend to predict

the label that occurs most often in the training set.

Building on these findings, we then created the E2, E3 and T5 model versions, i.e., new models for both ERA-Interim and TRMM (as well as a combination of them) that were primarily a repetition of the best previous experiments with updated labels for the new onset dates.

Data preprocessing Due to the new distribution of onset dates, the prediction date was newly set on the 22nd of May, increasing the available sequence length by 11 days (up to 83 days). The preprocessing otherwise didn't change much, as the TRMM dataset could simply be regarded as a one-channel version of the stacked ERA-Interim features.

Modeling The models were not much changed either, as they were already flexible enough to accept arbitrary sequence lengths and an arbitrary number of stacked features (channels). The main changes were mostly refactorings and optimizations in the order of layers. As such, we do not explicitly show the source for these kinds of models.

Preliminary results Running the experiments for E2 and T5 yielded much better final losses. The validation loss could get as close as 2.5 days (TRMM) and 4.7 days (ERA) of mean-absolute error, which would suggest that T5 could predict the monsoon onset on the 22nd of May and do so with an accuracy of +- 2.5 days, which would have been a good result.

However, close examination of the actual predictions of the model was not as reassuring, as the model still seemed to predict only one single number for any validation or test example that was fed into the model. Furthermore, a combination of the TRMM and ERA dataset by stacking the three features yielded comparable results: the model again seemed to resort to predicting the mean of onsets based on the TRMM dataset. Additionally, stacking the TRMM and the ERA-Interim dataset necessitated that we excluded years before 1998 from the ERA-Interim dataset, causing a hefty loss of available training data.

The development of the E4-models

Following the bad generalization capabilities of all models up to their latest versions, an improvement to the overall training approach was sought. Up to versions E3 and T5, the dates of prediction were always fixed to a single date during the pre-monsoon period (22nd of May). This meant that for each year of data, only a single sequence could be extracted and used as a training example.

The patterns during the pre-monsoon period might not be conclusive enough to allow a model to predict future onsets reliably. Some patterns might be closely tied to a temporal distance to the monsoon onset, i.e., some event might always happen approximately two weeks before onset. This would have been captured by the model at different steps in the sequence, which could, in turn, have been suitable for prediction. On the other hand, the model might have missed many of these events because the data was cut off at a fixed prediction date, possibly multiple weeks before the real onset date.

To allow a model to capture all the events leading up to the onset of each respective year, the approach to training the model thus needed to be changed. Firstly, the model needed to be fed sequences with equal distances to the respective real onset date. For example, with an onset on the 10th of June, the sequence to be fed to the model could include data from the 9th of May to the 9th of June. Repeating this for each year in the training set while keeping a one-day distance

to the real onset date and the length of the sequences fixed, the model should learn to predict that the onset will take place in exactly one day.

This would not be that useful by itself, as the model would then only be able to predict the onset one day in advance. However, this approach to training allows feeding the model multiple sequences per year, each with a different offset to the onset date and a correspondingly different label. The amount of available examples is as such greatly increased: generating three such sequences per year would triple the overall number of examples available for training. Furthermore, the prediction capabilities of the model could then be evaluated separately for each offset, yielding results on how well the model can predict one day in advance, one week in advance or even multiple weeks in advance.

The detailed architecture and evaluation of this final model architecture have already been described in depth in previous sections (Section 4.3.1, especially). Having described all the models that we used during the creation of this work, we now conclude this chapter in the next section, and our work in the section after that.

4.4 Conclusion

In this second part of our work, we have described the different approaches to monsoon onset prediction that we have evaluated, along with their results. Next to many experiments that we would consider failures based simply on their prediction results, we have found a model architecture that can learn from the ERA-Interim dataset and accurately predicts the ISM onset from up to a one month lead. The architecture is based on the ConvLSTM network layers we have introduced early on and combines several of these layers with a sequence of fully-connected layers for further learning. A prediction is finally performed by a single-neuron output layer with linear activation.

While we have evaluated models based on different datasets for both input features and monsoon onset dates, we have found the ERA-Interim dataset to work best, especially when used in combination with IMD onset dates from Singh and Ranade (2009) and the IMD. The TRMM dataset was also evaluated but led to models that were unable to learn any useful patterns, which might have been caused by the natural sparsity of precipitation.

The accuracy achieved by our final model outperforms comparable methods like the official IMD prediction model (four days RMSE) or the approach presented in Stolbova (2015) with an accuracy of $\pm$ seven days. Over repeated evaluation of our overall best-performing model, the maximum mean absolute error of a repetition was 3.09 days, while the minimum was 2.46 days and the mean absolute error over all repetitions was 2.82 days.

5

Conclusion

The Indian Summer Monsoon is one of the most influential meteorological phenomena on earth. Even though the ISM occurs every year, the exact timing and strength of its occurrence are subject to great variability and a primary factor for the welfare of the population in countries all over the Indian subcontinent. In a first part of this work, we have sought to provide insight into the strength of monsoon rainfalls by analyzing the spatial distribution of extreme rainfall events. The second part was then dedicated to the problem of predicting the timing of ISM onset over Kerala. With both of these parts, we have tried to reduce the uncertainty that is caused by the variability of ISM behavior. This chapter serves as a final conclusion to our work, with the main findings and results summarized in Section 5.1 and possible extensions of our work described in Section 5.2.

5.1 Summary & Final Conclusions

Trying to provide more insight into the strength of the ISM, we have analyzed extreme monsoon rainfall using event synchronization methods and produced climate networks representing the spatial distribution of extreme rainfall events. We have then analyzed these climate networks with network measures like degree, betweenness centrality and PageRank, resulting in an intuition about the importance of locations concerning extreme rainfall. When compared to the foundational work of Stolbova (2015), we have identified similar key regions for the pre-monsoon and monsoon seasons: while the Indian Ocean and the Tibetan Plateau are the most influential regions before the onset of the ISM, North Pakistan dominates the climate networks during the monsoon season. Our results for the post-monsoon season differ from the results in the foundational work, as the important locations in our network are concentrated over central India and the Tibetan Plateau, while the results from Stolbova (2015) primarily focus on the Tibetan Plateau.

For all of our climate network results, we have tried to link the previously described patterns to well-known phenomena that drive the yearly development of the ISM. The pre-monsoon season corresponds to the Indian summer, in which the country regularly experiences temperatures of more than 40°Celsius. Additionally, most rainfall is concentrated over the oceans in this season. The monsoon season is characterized by a reversal of the land-ocean temperature gradient, leading to a sea-breeze that brings moisture from the oceans onto the land. While it is non-trivial to interpret a distinct pattern like the one over North Pakistan, we proposed that the region could serve as a pivot between other key regions during this season. The final monsoon season we have analyzed, the post-monsoon season, is the season where the ISM fully withdraws from

the subcontinent, and the following north-east monsoon starts to build up. We have proposed that this might cause patterns like the one seen over central India and the Tibetan Plateau.

While our first results were based mostly on the work of Stolbova (2015), we have extended her climate network analysis by the PageRank measure as well as weighted and directed climate networks and networks thresholded at different percentiles. The PageRank of directed networks shows some new patterns when compared to the degree, as the intuition of importance is different in its ranking procedure. Further, we have found that a weighting of network edges with their synchronization coefficient makes little difference in the overall patterns and only changes minimal details. While the results of Stolbova (2015) are based on the TRMM dataset at its full resolution (0.25°), we have reduced the dimensionality and thus the computational requirements of the dataset by aggregating the resolution to 0.75°. While this could also be a reason for some of the differences in our results, they largely seem to match for two out of three seasons, with only the third season displaying bigger differences.

The second main research question in this work was the prediction of ISM onset, for which we have built prediction models based on recent neural network methods. After a long process of parameter tuning and model building, we finally got to a model architecture that manages to predict ISM onset better than comparable methods and does so from up to a one month lead. Official IMD predictions achieve a root-mean-squared error of around four days, meaning that the monsoon onset prediction can be interpreted as accurate if the real onset occurs up to four days earlier or later than the predicted date. Stolbova (2015) even counts predictions as accurate if the real onset is up to seven days early or late. In comparison, repeated evaluation of our best model yields an overall mean absolute error of 2.82 days when predicting between one and thirty days before the monsoon onset. The mentioned model tends to predict best during the week leading up to the monsoon onset, as well as more than three weeks before the onset, but performance is still good in between these periods.

Our overall best-performing model architecture takes several features of the ERA-Interim reanalysis dataset as input (mean sea-level pressure, temperature and relative humidity). Based on these inputs, a sequence of ConvLSTM and fully-connected layers manages to perform an accurate numerical prediction. On any day during the month leading up to the ISM onset, we can use our model to predict the number of days that remain until the monsoon arrives. Predictions could even be performed every day, resulting in updated predictions as more data gets available.

Even though the predictions are quite accurate, an important shortcoming in this model architecture is the practicality of the prediction approach. We trained the majority of our neural network models based on the ERA-Interim dataset, which is a meteorological reanalysis computed from both observations and simulations. The data assimilation and quality control procedures of such reanalysis datasets take their time, which is why the ERA-Interim dataset is only published with a two-month delay (not in real-time). This means that the ERA-Interim dataset cannot practically be used to predict the ISM onset, as the data for the full pre-monsoon season is only released in July. However, our architecture still can still serve as a proof that onset prediction can be performed by applying neural networks to spatiotemporal meteorological datasets and could further be used as a foundation for future models.

5.2 Future Work

The work we have presented so far could be extended in several ways, of which some would require more effort than others. In this final part of our work, we introduce the possible future extensions we could imagine and state some of our ideas for their implementation.

Prediction of extreme rainfall events The analysis of climate networks as shown in this work is only a starting point, after which a next logical step could be the prediction of extreme rainfall events. For example, one could try to predict future extreme events at a location basing on extreme events in strongly synchronous locations (that tend to lead said location). This approach could probably only be used for short-term prediction, as the lead-time of synchronous events tends to lie within only a few days. However, even short-term predictions could potentially make a big difference concerning the safety of people affected by the predicted event.

Usage of different datasets Our final prediction model is fully focused on the ERA-Interim dataset and several of its features. While the predictions based thereon seem to be accurate and manage to outperform other comparable models, the practicality of a model based on such a reanalysis dataset is not as high as one would wish. Reanalysis datasets like ERA-Interim are only released after a certain delay (i.e., two months for ERA-Interim), which is needed for the producers to be able to perform data assimilation and quality control procedures. However, this also means that most pre-monsoon data is only fully available once the monsoon has most certainly already arrived. We suggest that the models we proposed could serve as a foundation for models that directly base on real-time datasets or purely observational datasets, and possibly multiple of them.

Usage of climate network results for onset prediction As of now, the two parts of this work, the evaluation of climate networks and the prediction of ISM onset, are strictly separate. However, a combination of the two could be imaginable, where a network measure like PageRank could be used as an additional input feature for neural network models. Such an additional feature might provide weights for the locations on the grid, which could help a neural network identify "important" locations that might work well as a predictor of monsoon onset.

Dense prediction Our work is solely focused on the prediction of the monsoon onset over Kerala, as it marks the arrival of monsoon on the Indian subcontinent. However, as we have described early on, an accurate estimate of monsoon arrival is of high importance for people all over India. Therefore, it would be much more useful if accurate predictions were available for all regions of India.

As a future improvement of the general applicability of our neural network models, we suggest the evaluation of an approach using "dense prediction" methods. Dense prediction models predict a label for each unit of the input, where a unit could be a single pixel or a region of the input image. A similar approach has been successfully applied to problems like video frame prediction, where, based on the past few frames, the next frame is to be predicted. The original work describing the ConvLSTM neural network architecture has in a sense also performed a dense prediction task, as future radar maps have been predicted based on past radar maps.

When applied to the problem of monsoon onset prediction, we could imagine that a future model would be trained to predict matrices containing the days until ISM onset for different patches of the Indian subcontinent. As we have successfully used the IMD onset dates as extracted from Singh and Ranade (2009), we would suggest a matrix with predictions based on the subregions described therein (as shown in Fig. 5.1). A suitable approach for handling patches of the input that are not part of any of the subregions would still need to be found, as the output dimensions of dense prediction tasks are typically equal to the input dimensions.

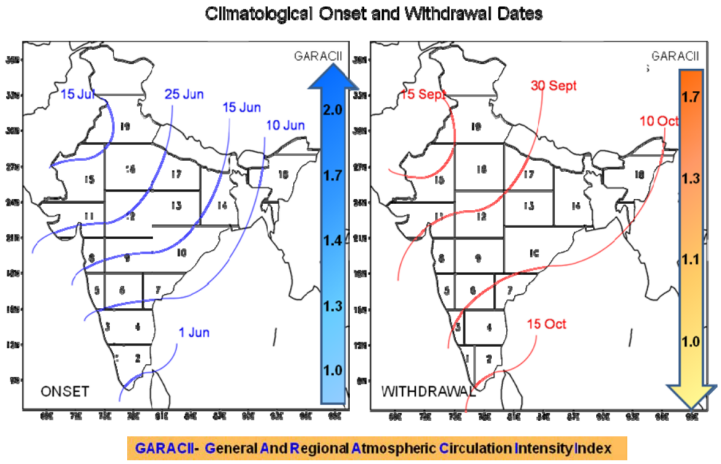

Figure 5.1: Regular monsoon onset and withdrawal dates for the 19 subregions of the Indian subcontinent as depicted in Singh and Ranade (2009).

Prediction of monsoon withdrawal While we have focused on predicting the monsoon onset, the withdrawal of monsoon is of similarly high importance to the Indian population, as it marks the transition from the rainy season to the winter months. Basing on our final model architecture, the withdrawal dates of the ISM could be predicted without necessitating any large changes. Using data for the monsoon season as extracted from ERA-Interim and onset dates from the IMD and Singh and Ranade (2009), a well-tuned model should be able to predict monsoon withdrawal with reasonable accuracy.

References

Auffhammer, M., Ramanathan, V., & Vincent, J. R. (2012). Climate change, the monsoon, and rice yield in india. *Climatic Change*, *111*(2), 411–424. doi: 10.1007/s10584-011-0208-4

Brandes, U. (2001). A faster algorithm for betweenness centrality*. *The Journal of Mathematical Sociology*, *25*(2), 163–177. doi: 10.1080/0022250X.2001.9990249

Central Intelligence Agency. (05.01.2018). *The world factbook.* Retrieved 09.01.2018, from https://www.cia.gov/library/publications/the-world-factbook/geos/in.html

Dee, D. P., Uppala, S. M., Simmons, A. J., Berrisford, P., Poli, P., Kobayashi, S., ... Vitart, F. (2011). The era-interim reanalysis: configuration and performance of the data assimilation system. *Quarterly Journal of the Royal Meteorological Society*, *137*(656), 553–597. doi: 10.1002/qj.828

Goddard Earth Science Data Information and Services Center. (2011). *Global precipitation measurement (gpm) mission overview.* Retrieved 26.12.2017, from https://pmm.nasa.gov/GPM

Goddard Earth Science Data Information and Services Center. (2016). *Trmm (tmpa) precipitation l3 1 day 0.25 degree x 0.25 degree v7.* Retrieved 26.12.2017, from https://disc.gsfc.nasa.gov/datacollection/TRMM_3B42_Daily_7.html

Huffman, G. (2016). *The transition in multi-satellite products from trmm to gpm (tmpa to imerg).* Retrieved 26.12.2017, from https://pmm.nasa.gov/sites/default/files/document_files/TMPA-to-IMERG_transition_161025.pdf

Huffman, G., & Bolvin, D. T. (2017). *Trmm and other data precipitation data set documentation.* Retrieved 26.12.2017, from https://pmm.nasa.gov/sites/default/files/document_files/3B42_3B43_doc_V7_4_19_17.pdf

Huffman, G. J., Pendergrass, A., & National Center for Atmospheric Research Staff. (2017). *The climate data guide: Trmm: Tropical rainfall measuring mission.* Retrieved 26.12.2017, from https://climatedataguide.ucar.edu/climate-data/trmm-tropical-rainfall-measuring-mission

India Meteorological Department. (2017). *Website.* Retrieved from http://www.imd.gov.in/Welcome%20To%20IMD/Welcome.php

Jin, Q., & Wang, C. (2017). A revival of indian summer monsoon rainfall since 2002. *Nature Climate Change*, *7*(8), 587–594. doi: 10.1038/nclimate3348

Malik, N., Marwan, N., & Kurths, J. (2010). Spatial structures and directionalities in monsoonal precipitation over south asia. *Nonlinear Processes in Geophysics*, *17*(5), 371–381. doi: 10.5194/npg-17-371-2010

Ordoñez, P., Gallego, D., Ribera, P., Peña-Ortiz, C., & García-Herrera, R. (2016). Tracking the indian summer monsoon onset back to the preinstrument period. *Journal of Climate*, *29*(22), 8115–8127. doi: 10.1175/JCLI-D-15-0788.1

Page, L., Brin, S., Motwani, R., & Winograd, T. (1999). *The pagerank citation ranking: bringing order to the web.*

Paliwal, A. (24.09.2017). Why india struggles to predict the weather over its lands. *The Wire*. Retrieved 21.01.2018, from https://thewire.in/180850/imd-weather-prediction-forecast-monsoons-drought-agrometeorology-kharif/

Pradhan, M., Rao, A. S., Srivastava, A., Dakate, A., Salunke, K., & Shameera, K. S. (2017). Prediction of indian summer-monsoon onset variability: A season in advance. *Scientific reports*, *7*(1), 14229. doi: 10.1038/s41598-017-12594-y

Quian Quiroga, R., Kreuz, T., & Grassberger, P. (2002). Event synchronization: a simple and fast method to measure synchronicity and time delay patterns. *Physical review. E, Statistical, nonlinear, and soft matter physics*, *66*(4 Pt 1), 041904. doi: 10.1103/PhysRevE.66.041904

Shi, X., Wang, H., Gao, Z., Lausen, L., Yeung, D.-Y., Woo, W.-c., ... Chen, Z. (2015). *Convolutional lstm network: A machine learning approach for precipitation nowcasting*.

Singh, N., & Ranade, A. A. (2009). Determination of onset and withdrawal dates of summer monsoon across india using ncep/ncar re-analysis. doi: 10.13140/RG.2.1.2059.5367

Stolbova, V. (2015). *Indian summer monsoon* (Dissertation).

Sutskever, I., Vinyals, O., & Le, Q. V. (2014). Sequence to sequence learning with neural networks. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, & K. Q. Weinberger (Eds.), *Advances in neural information processing systems 27* (pp. 3104–3112). Curran Associates, Inc. Retrieved from http://papers.nips.cc/paper/5346-sequence-to-sequence-learning-with-neural-networks.pdf

Willetts, P. D., Turner, A. G., Martin, G. M., Mrudula, G., Hunt, K. M. R., Parker, D. J., ... Brooks, M. E. (2017). The 2015 indian summer monsoon onset - phenomena, forecasting and research flight planning. *Weather*, *72*(6), 168–175. doi: 10.1002/wea.2819

World Meteorological Organization. (2005). *The global monsoon system: Research and forecast* (No. 1266). Geneva, Switzerland.

Yihui, D., & Chan, J. C. L. (2005). The east asian summer monsoon: an overview. *Meteorology and Atmospheric Physics*, *89*(1-4), 117–142. doi: 10.1007/s00703-005-0125-z

A

Appendix

A.1 Structure of the Code Repository

- `code` — Source code of our implementation
  - `00_ARCHIVE` — Archived resources from early experiments and analyses
  - `00_CACHE` — Cached resources like calculated event sync matrices
  - `00_DATA`
    - `ERA` — ERA-Interim dataset and download script
    - `TRMM` — TRMM dataset and download script
    - `onset_dates_v2.csv` — Onset dates as extracted from Singh
  - `00_LOGS` — Network training logs for analysis with tensorboard
  - `01_ANALYSIS` — Source for climate network analysis
    - `Dataset.py` — Base class for both ERA and TRMM datasets
    - `ERA.py` — Class for ERA-Interim processing
    - `TRMM.py` — Class for TRMM processing
    - `Visualization.py` — Class for creation of visualizations (cartopy)
    - `ERA & TRMM.ipynb` — Notebook with feature analysis
    - `Onset Dates.ipynb` — Notebook with onset date analysis
    - `TRMM*.ipynb` — Notebooks with TRMM climate network analysis
  - `02_MODELLING` — Source for onset prediction models
    - `LSTM_E4.py` — E4-Script with all configurations for the 45 basic experiments
    - `LSTM_E4-final.py` — E4-Script with the final repeated configurations
    - `models` — Classes for the neural network models
  - `03_EVALUATION` — Evaluation results of neural network models
    - `histories` — Training histories of experiments (csv files with loss/vloss)
    - `logs` — Full training logs of experiments
    - `Accuracy Distributions.ipynb` — Notebook with final model accuracy analysis
  - `*.sh` — Shell scripts used for SLURM or PS experiments
- `resources` — Conda environment exports containing all important packages
  - `conda-*.env` — Conda environments for windows pcs
  - `gru-*.env` — Conda environments for linux machines
- `thesis` — Source code of the thesis and additional resources
  - `references`
    - `Experiments.xlsx` — Results of all TRMM and ERA experiments
    - `TestAccuracy.csv` — Predictions of all five final evaluations
    - `Visualizations.pptx` — Visualizations as found in the thesis

A.2 Monsoon Onset Dates

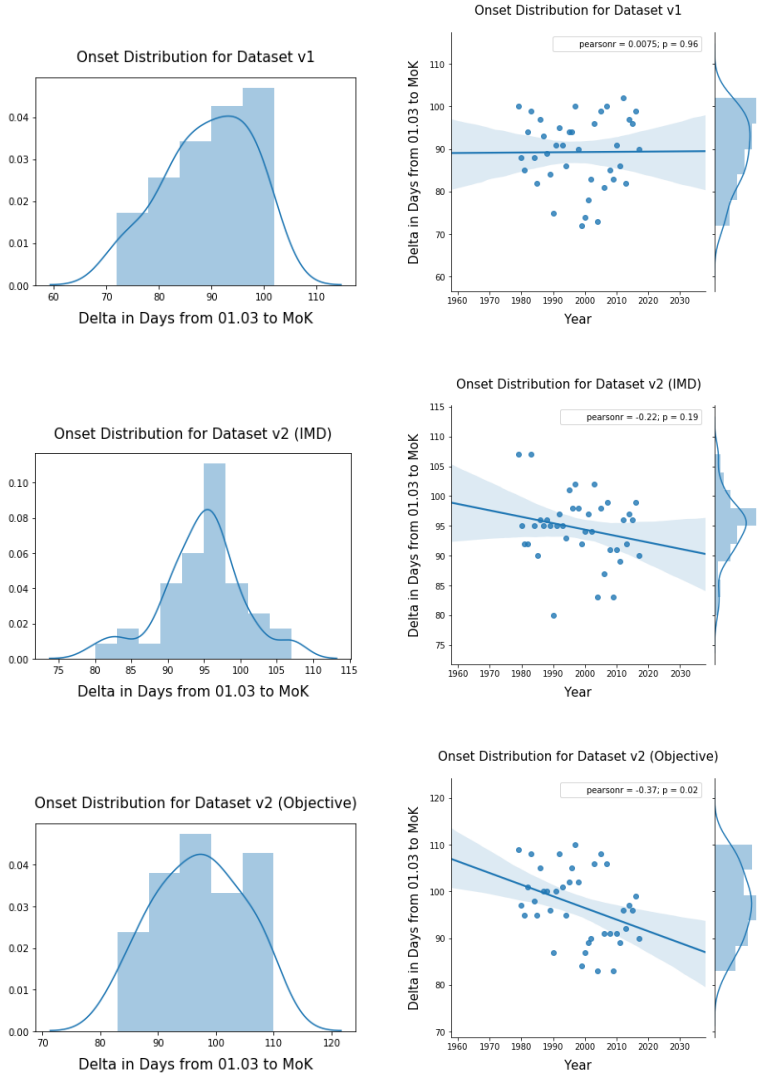

Figure A.1: Overview of Onset Distributions (1979-2017, v1: India Meteorological Department (2017); Ordoñez et al. (2016), v2: India Meteorological Department (2017); Singh and Ranade (2009)

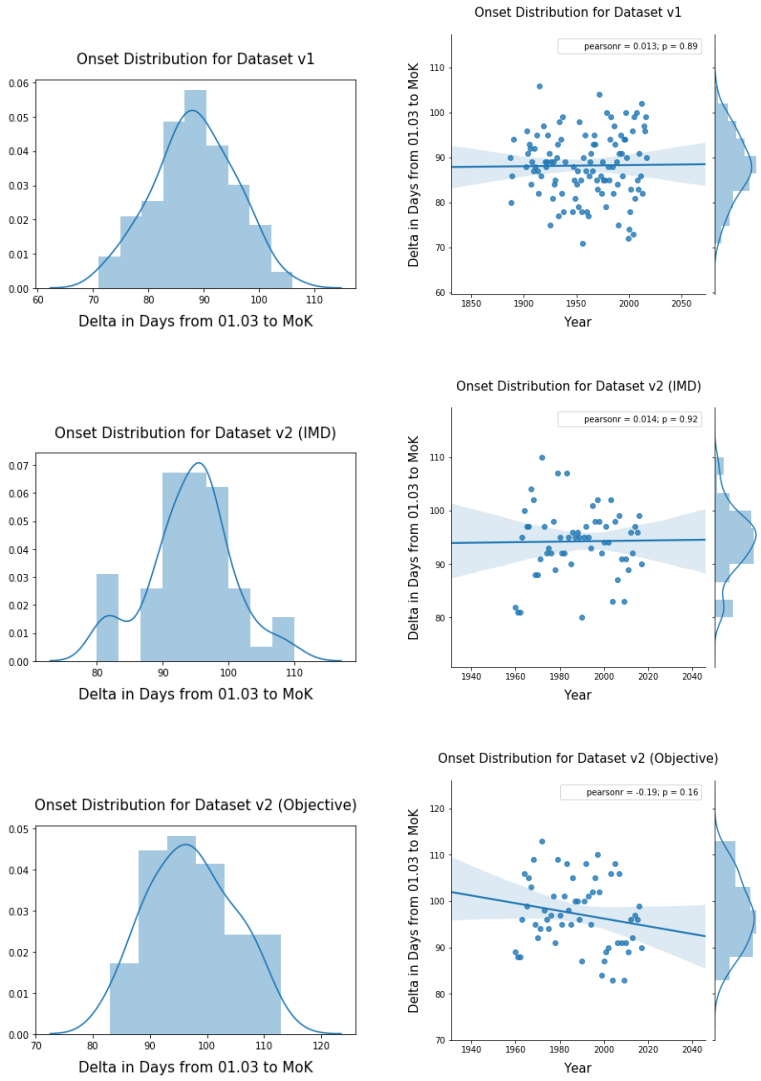

Figure A.2: Overview of Onset Distributions (v1: 1887-2017, India Meteorological Department (2017); Ordoñez et al. (2016), v2: 1960-2017, India Meteorological Department (2017); Singh and Ranade (2009))

A.3 TRMM & ERA

A.3.1 Getting the datasets

The process for downloading the TRMM dataset can be summarized as follows:

1. Create an account for NASA Earthdata at `https://urs.earthdata.nasa.gov/home`
2. Subset the TRMM dataset using `https://disc.gsfc.nasa.gov/SSW/#keywords=TRMM_3B42_Daily%207`
3. Select the *precipitation* variable (the aggregation of the other ones)
4. Download the list of URLs in a file as offered in the subset results
5. To download the dataset, follow the official instructions (especially setting the cookie) as offered in `https://disc.gsfc.nasa.gov/SSW/SSW_URL_List_Downloading_Instructions.html` (can be used in combination with our download script)

The process for downloading of the ERA-Interim dataset is slightly more complex, as it requires the usage of an official library to download more than one month at a time. While the dataset could also be downloaded using `http://apps.ecmwf.int/datasets/data/interim-full-daily/levtype=sfc/`, this subset wizard only allows the download of one month at a time. The full process is structured as follows:

1. Create an account for ECMWF data access at `https://apps.ecmwf.int/registration/`
2. Follow the description as can be found on `https://software.ecmwf.int/wiki/display/WEBAPI/Access+ECMWF+Public+Datasets`
3. Our download script is already setup to download all the ERA-Interim features used in this work. However, it assumes that dataset access has already been setup.

A.3.2 Usage with Python

When it comes to handling meteorological datasets, there are two data formats one needs to know of. These formats, GRIB and NetCDF, store the datasets using optimized binary files and one needs a compatible library to read them. For NetCDF, such libraries are easy to find. The one we have found to work best is *xarray*, which can be found at `http://xarray.pydata.org/en/stable/io.html`.

The xarray library would also allow reading in GRIB files when the additional *PyNIO* library is installed. However, we have not been able to get PyNIO to work (or any other GRIB-related Python library, at that). TRMM and ERA-Interim both support the download of data using NetCDF and we would strongly suggest using this format whenever possible (at least when using Python).

A.3.3 Feature analysis

The visualizations in this section show the development of different relevant ERA and TRMM features over the duration of an average monsoon. To calculate the visualizations, the data for each feature has been averaged over all years available (1998-2017 for TRMM, 1979-2017 for ERA).

Example: The temperature grids 10 days previous to the objective onset dates (MoK; A.6) for all available years are extracted and then averaged to yield an averaged grid representation. The visualization is then simply calculated based on the averaged grid.

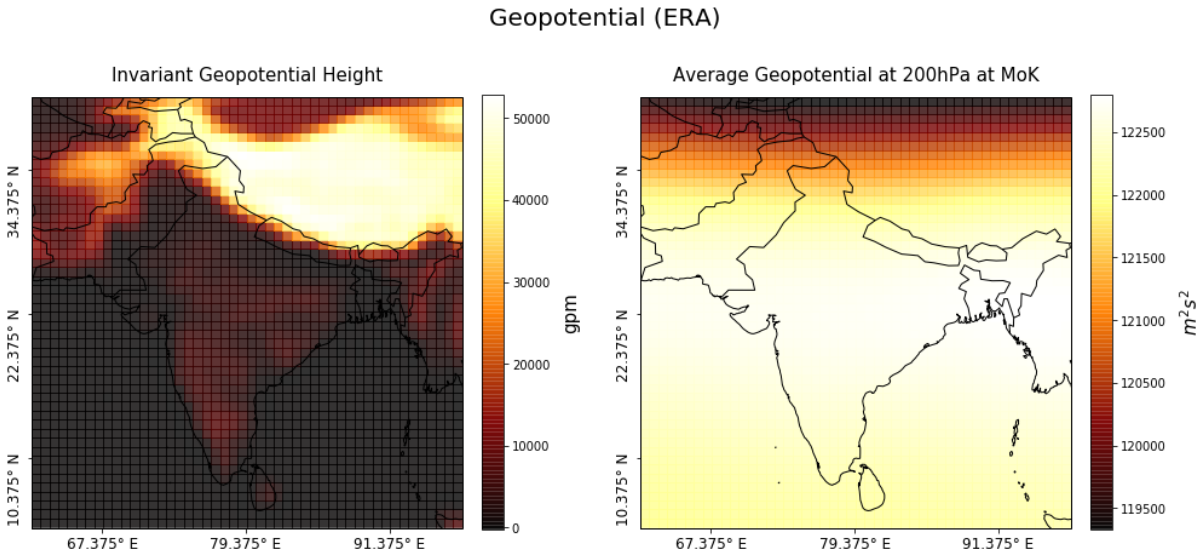

Figure A.3: Geopotential Height and Geopotential at 200hPa (ERA-Interim, 1979-2017)

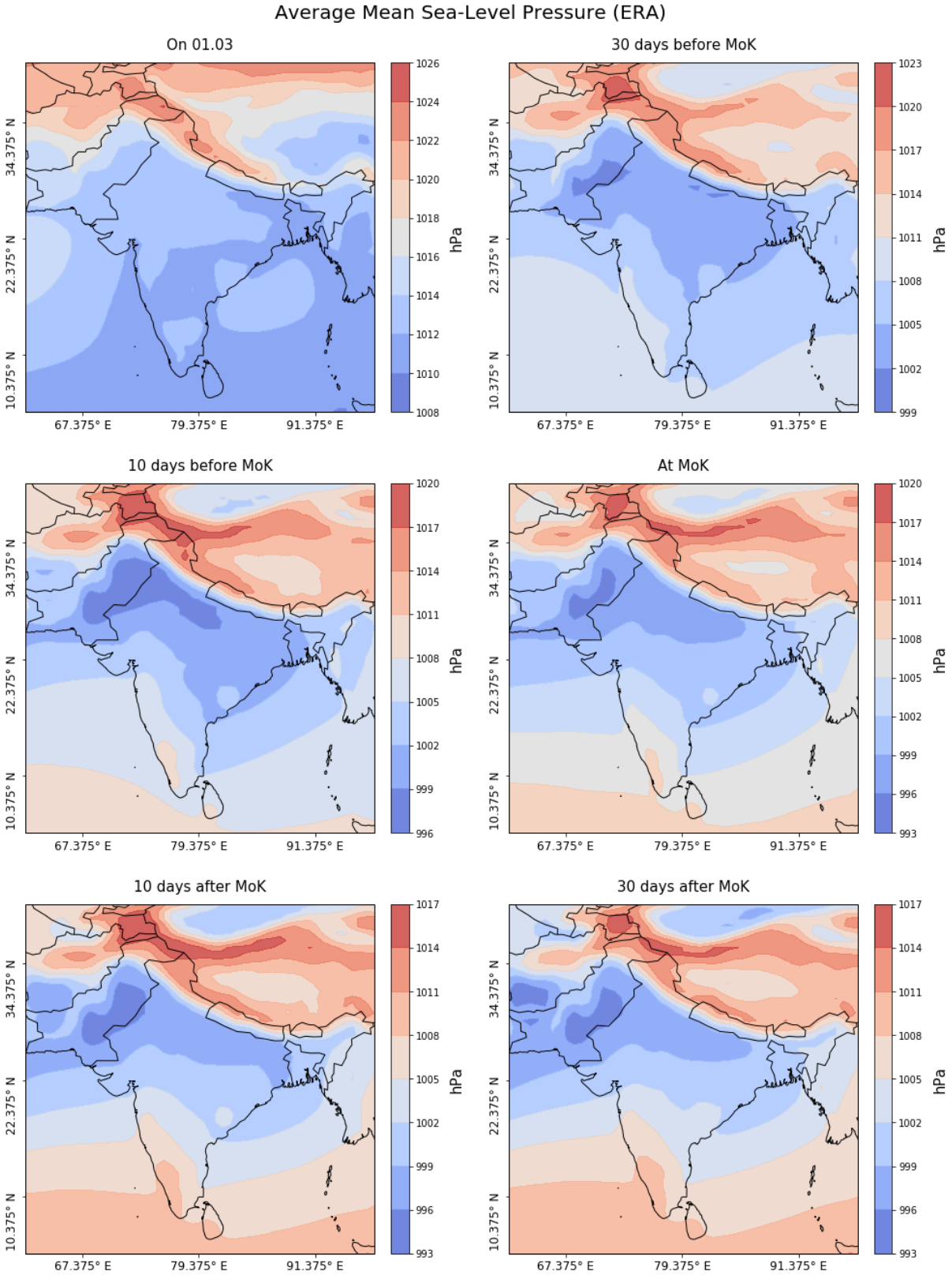

Figure A.4: Average Mean Sea-Level Pressure (ERA-Interim, 1979-2017)

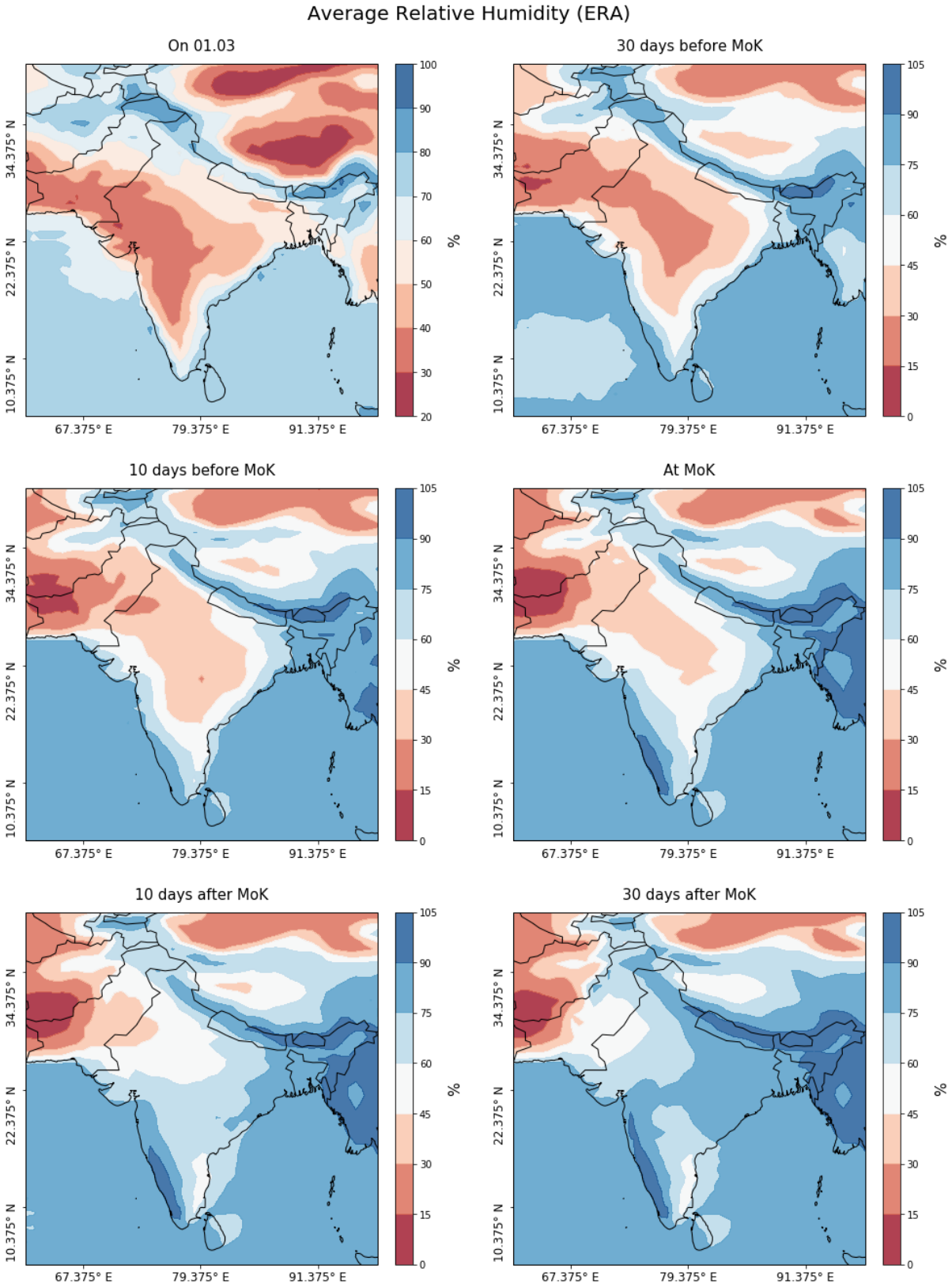

Figure A.5: Average Relative Humidity at 1000hPa (ERA-Interim, 1979-2017)

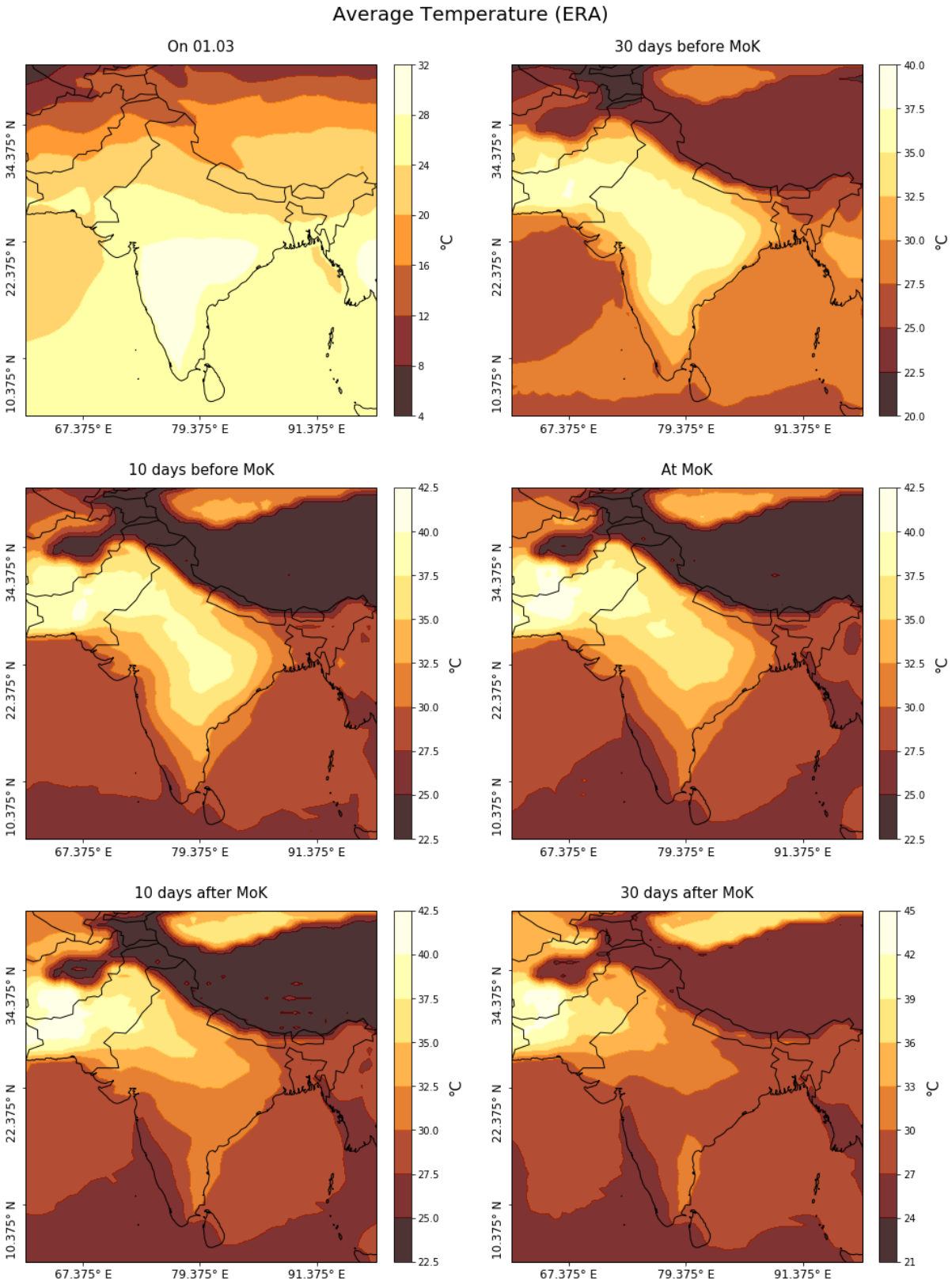

Figure A.6: Average Temperature at 1000hPa (ERA-Interim, 1979-2017)

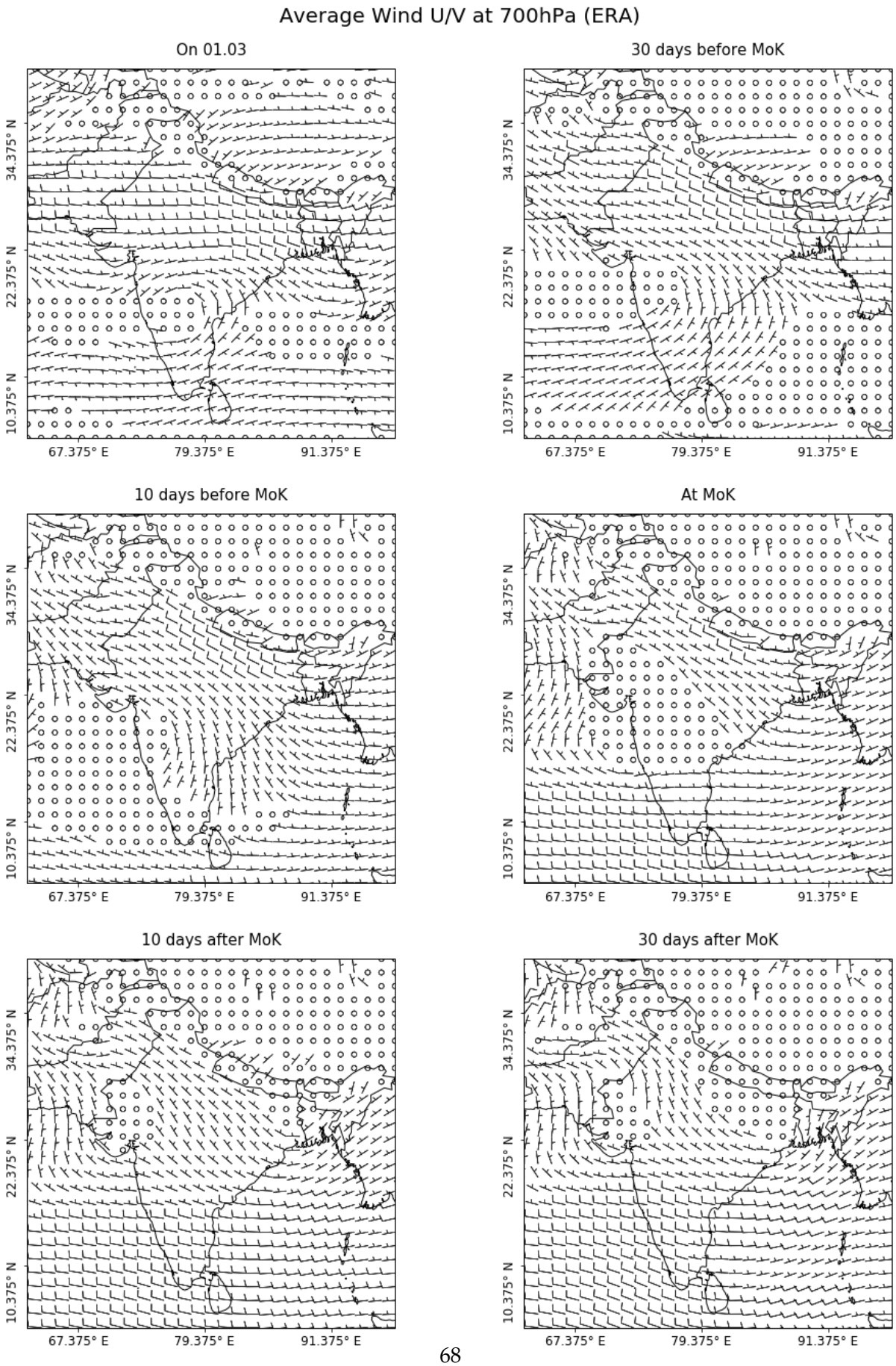

Figure A.7: Average Wind (U/V) at 700hPa (ERA-Interim, 1979-2017)

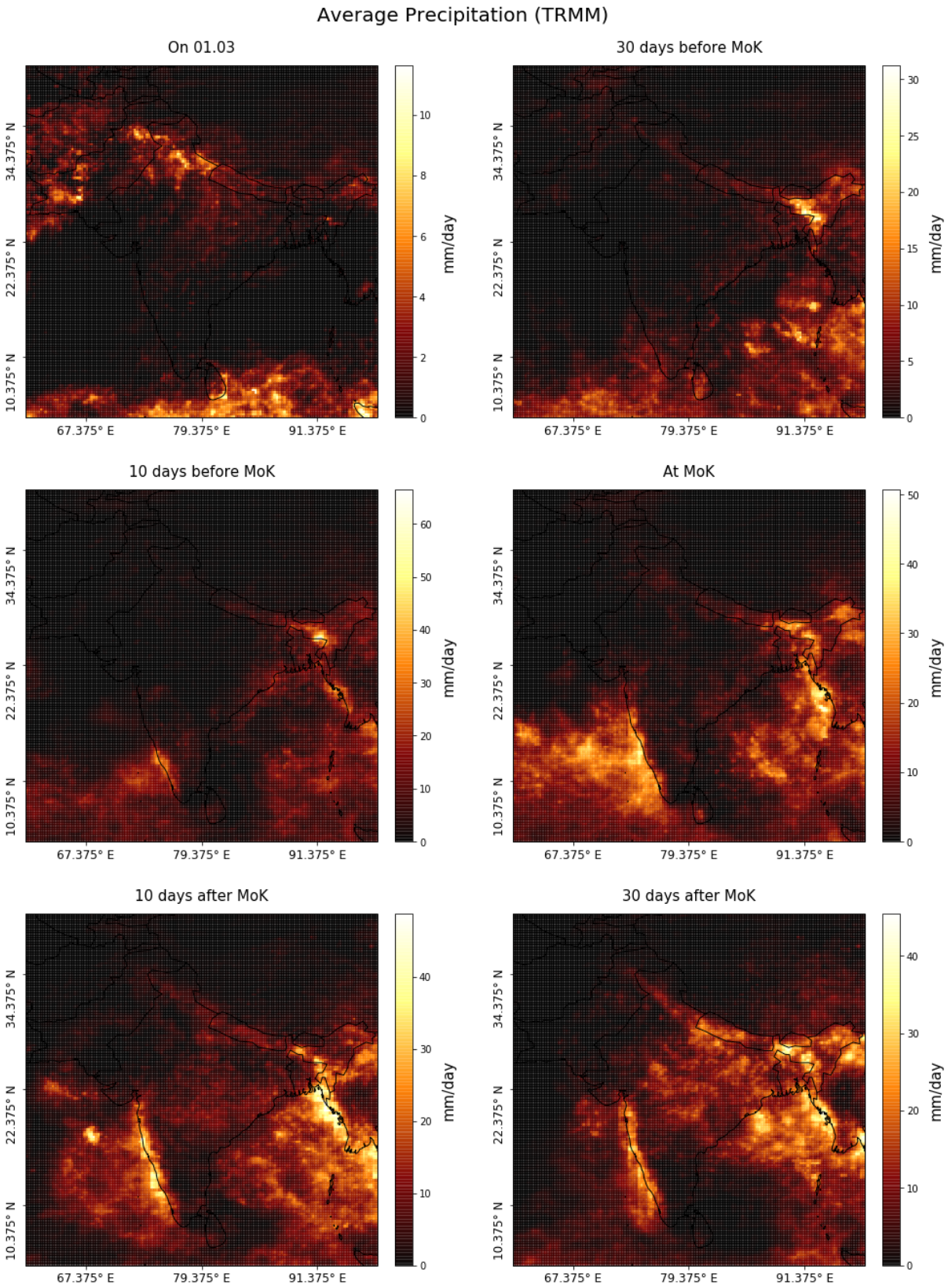

Figure A.8: Average Precipitation (TRMM, 1998-2017)

A.4 Event Synchronization

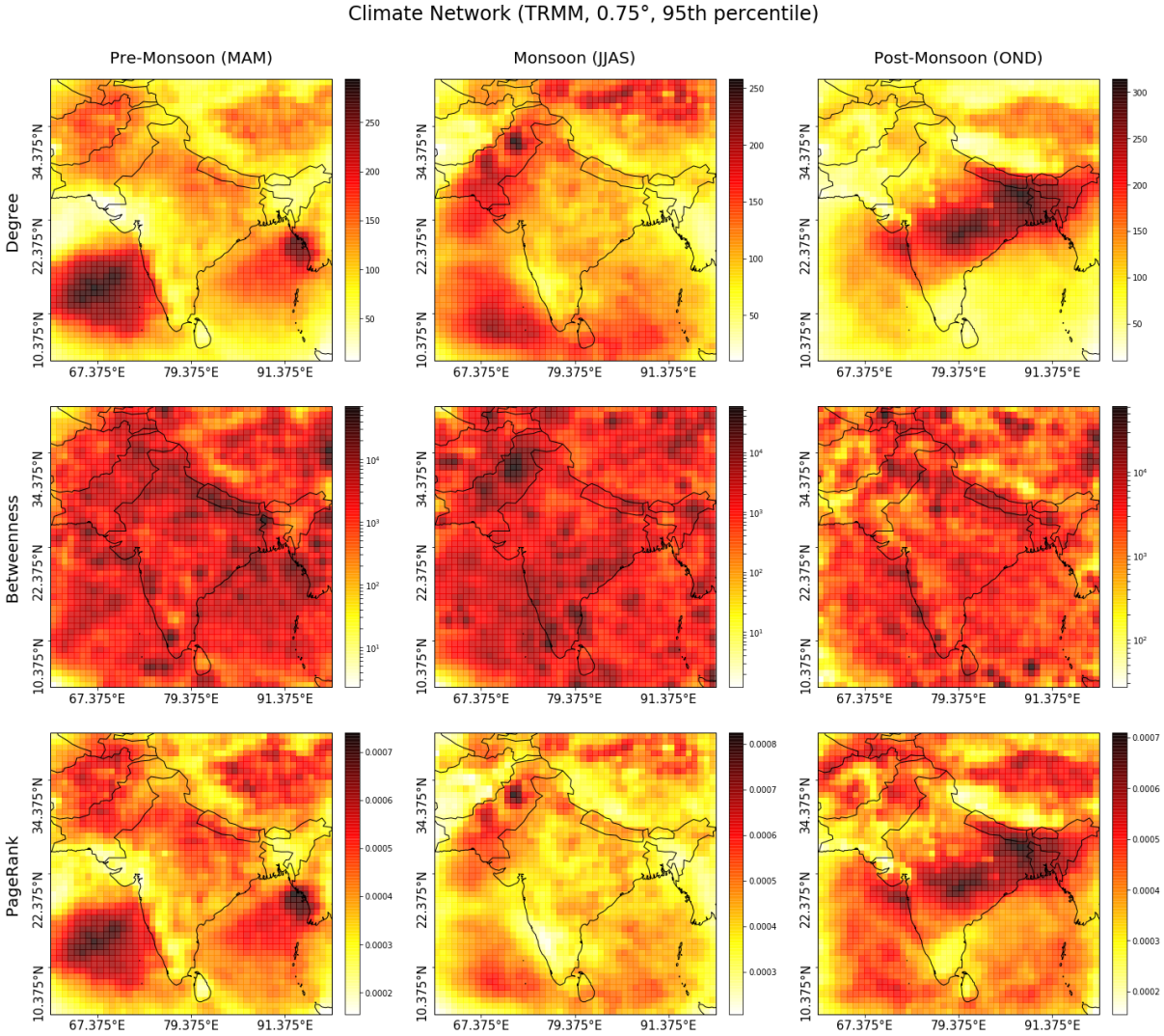

Figure A.9: Climate network analysis for unweighted, undirected climate networks (based on TRMM at 0.75° resolution and a threshold set at the 95th percentile).

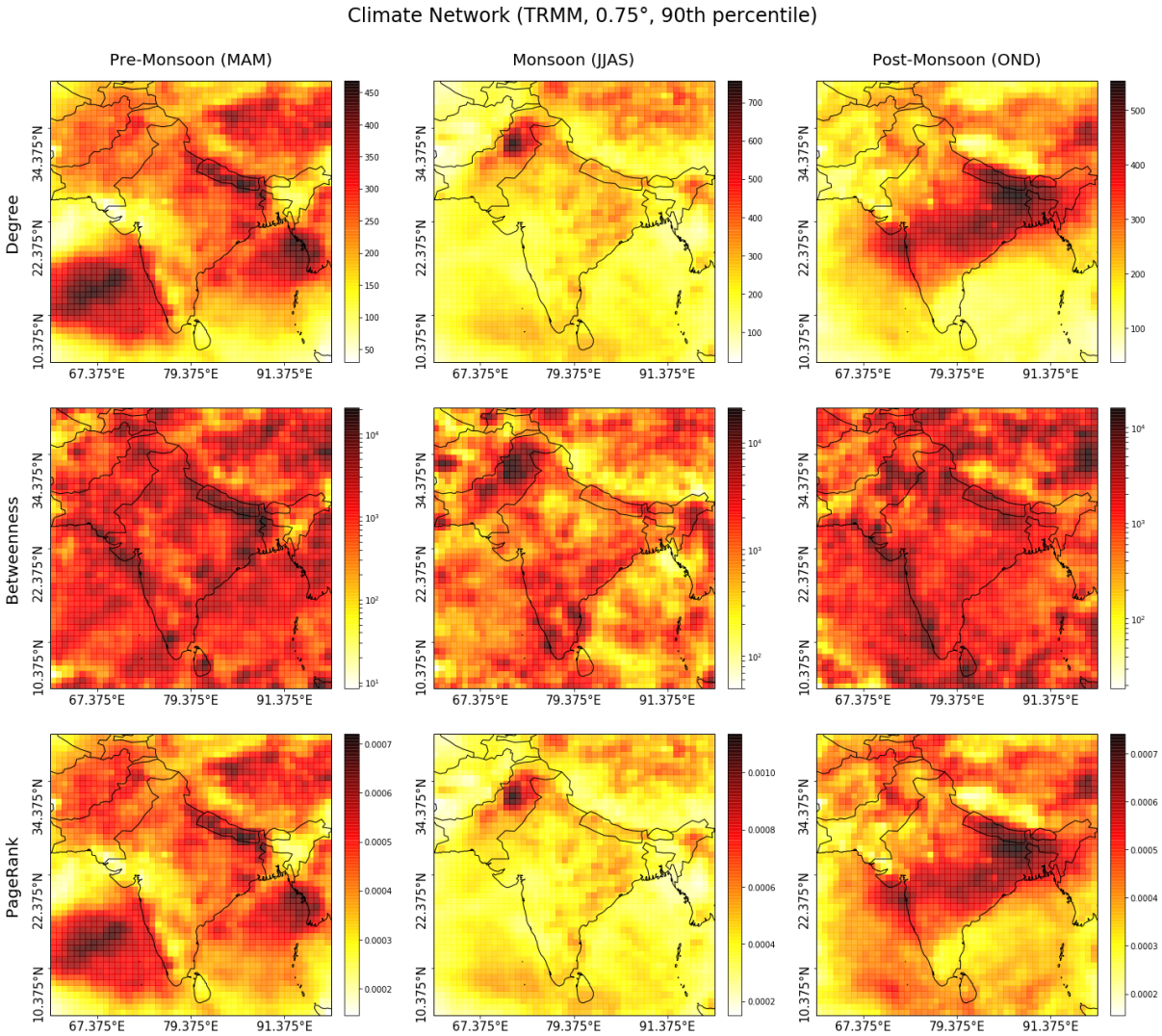

Figure A.10: Climate network analysis for unweighted, undirected climate networks (based on TRMM at 0.75° resolution and a threshold set at the 90th percentile).

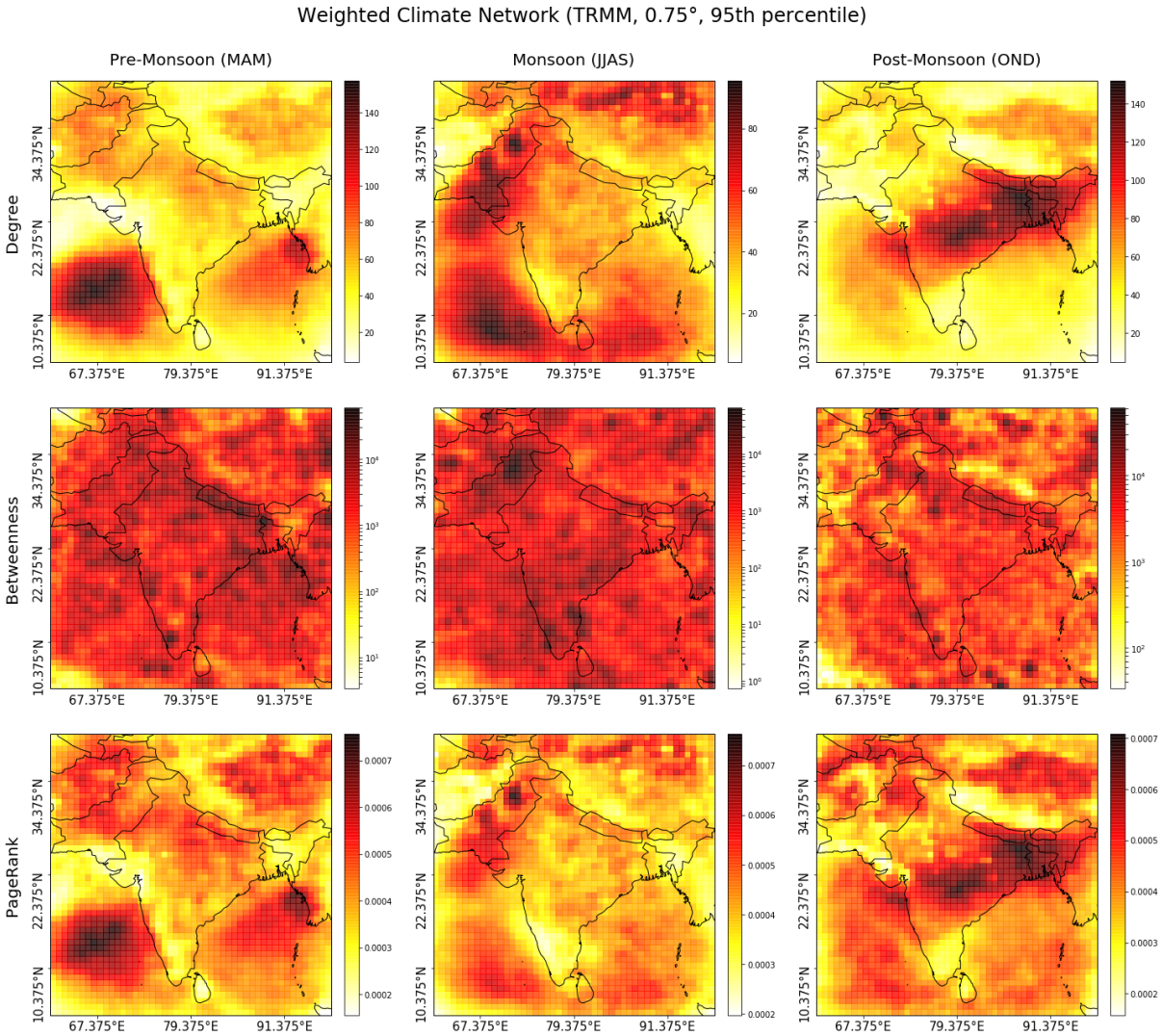

Figure A.11: Climate network analysis for weighted, undirected climate networks (based on TRMM at 0.75° resolution and a threshold set at the 95th percentile).

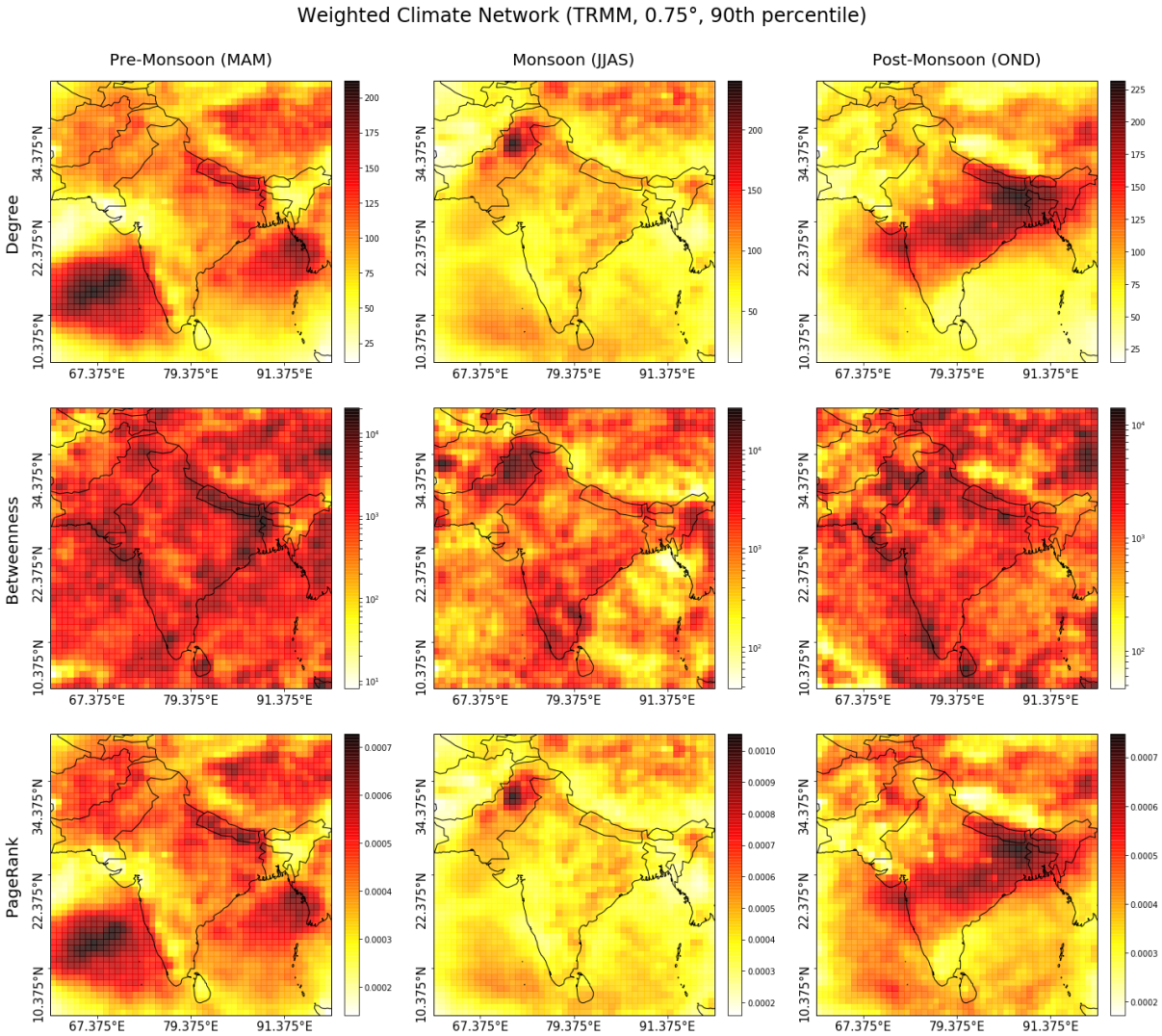

Figure A.12: Climate network analysis for weighted, undirected climate networks (based on TRMM at 0.75° resolution and a threshold set at the 90th percentile).

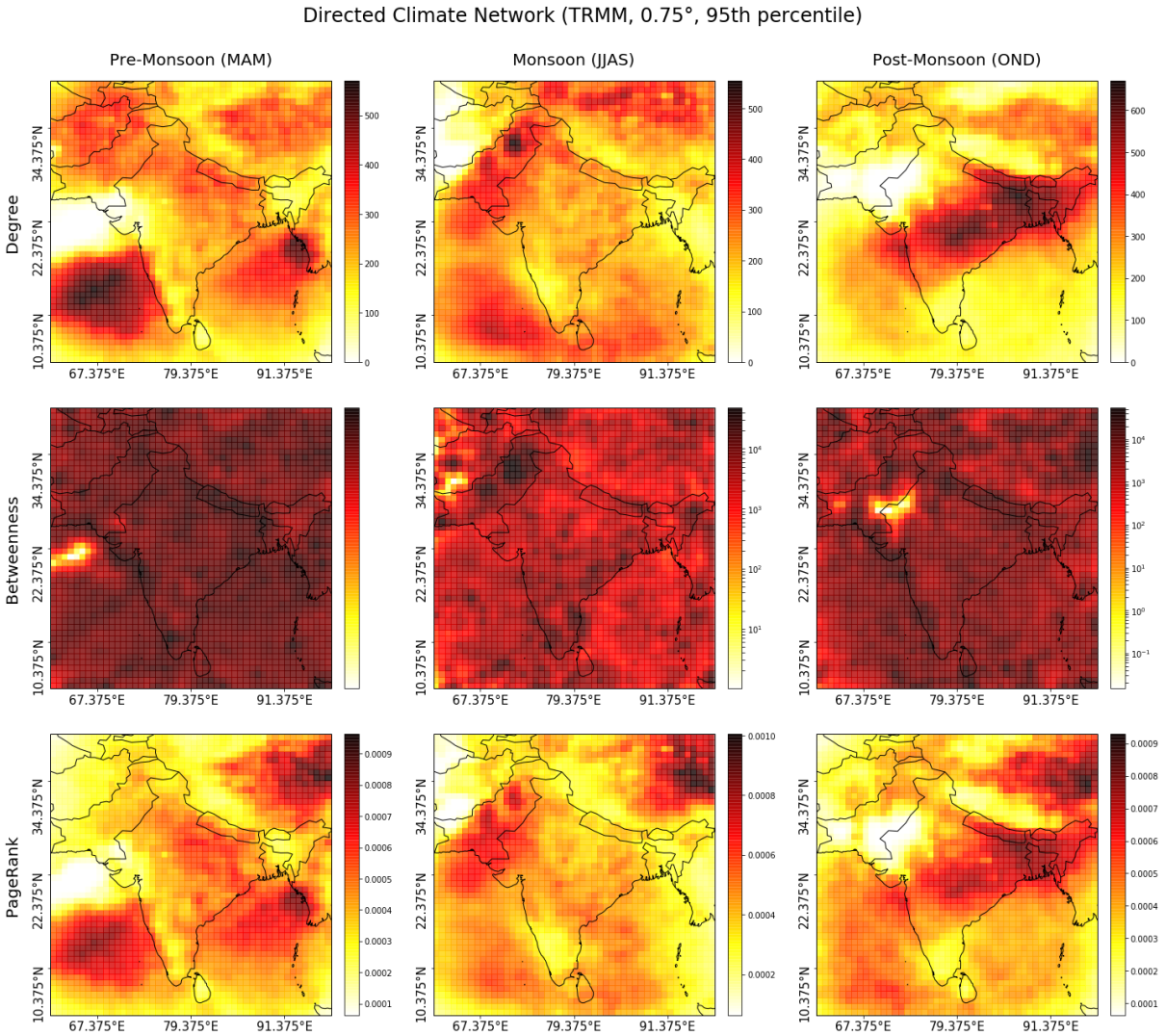

Figure A.13: Climate network analysis for unweighted, directed climate networks (based on TRMM at 0.75° resolution and a threshold set at the 95th percentile).

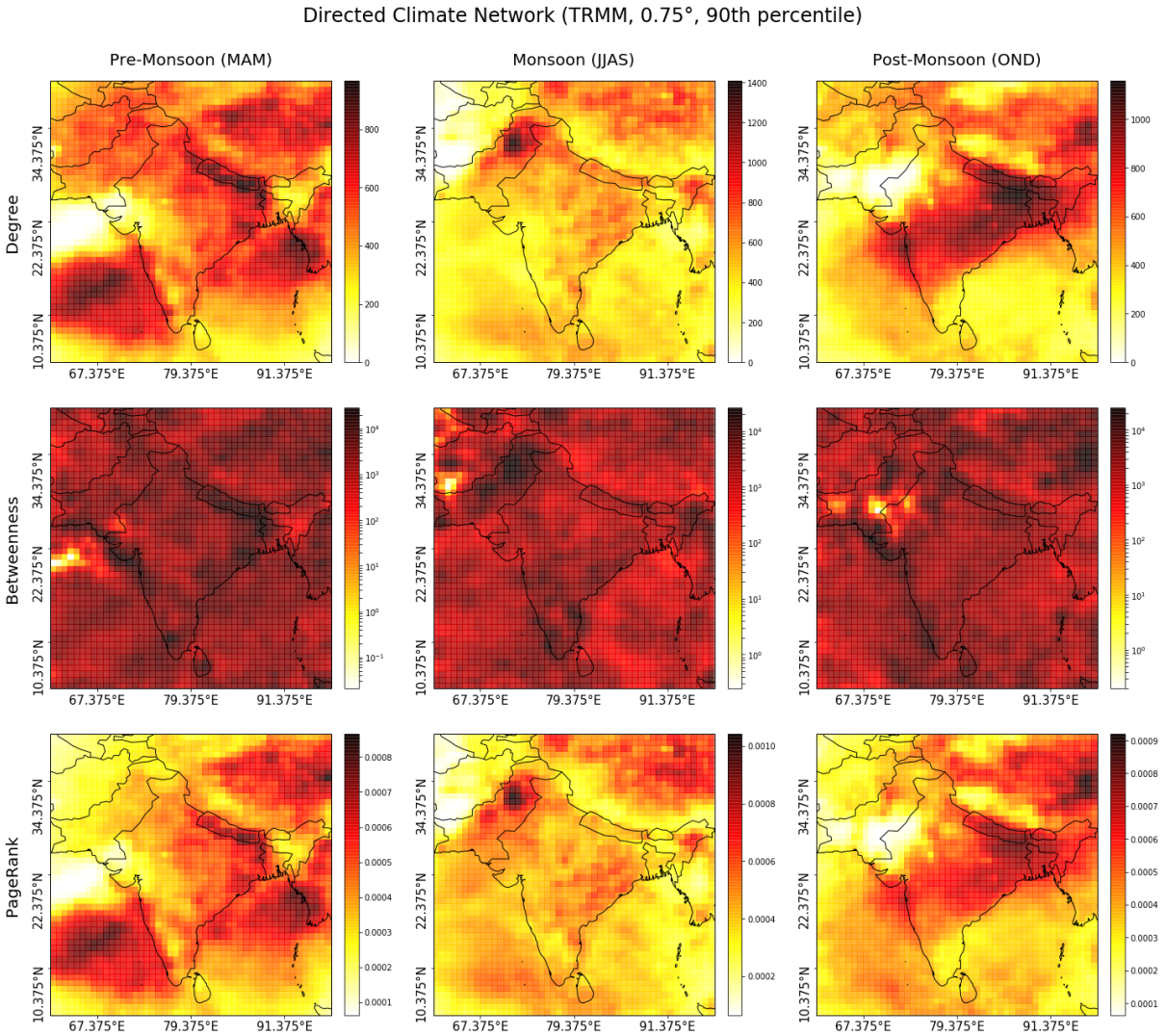

Figure A.14: Climate network analysis for unweighted, directed climate networks (based on TRMM at 0.75° resolution and a threshold set at the 90th percentile).

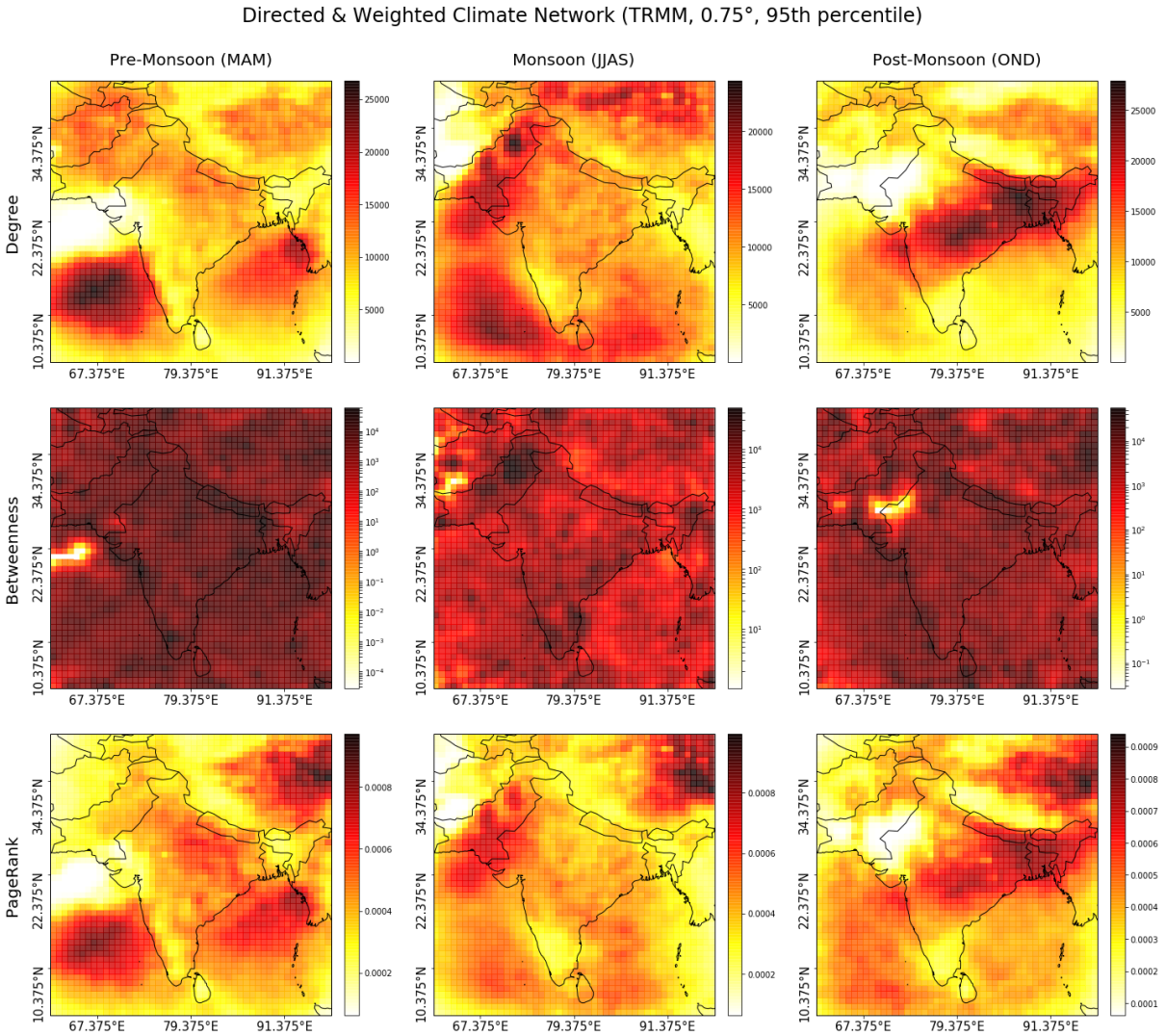

Figure A.15: Climate network analysis for weighted, directed climate networks (based on TRMM at 0.75° resolution and a threshold set at the 95th percentile).

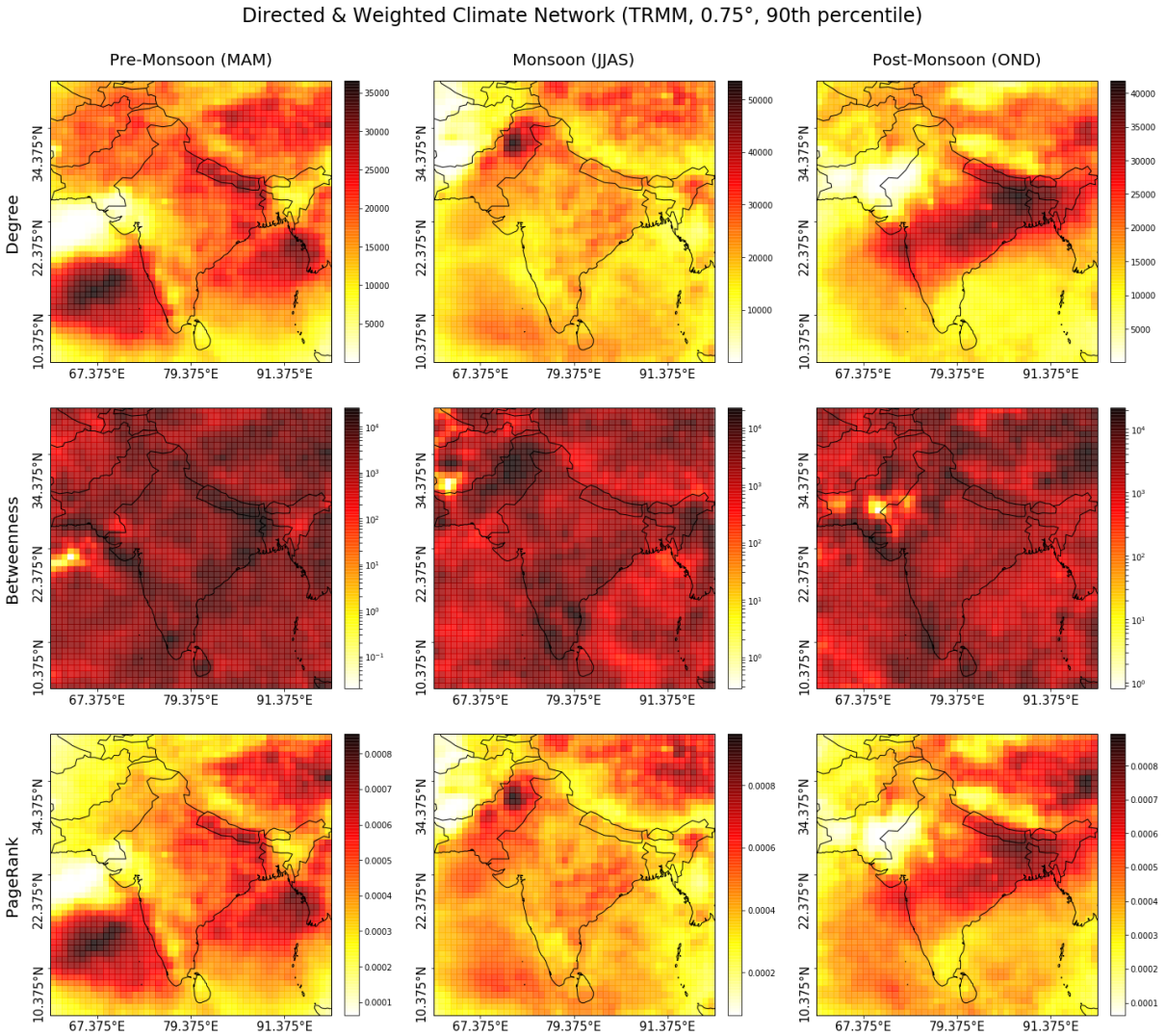

Figure A.16: Climate network analysis for weighted, directed climate networks (based on TRMM at 0.75° resolution and a threshold set at the 90th percentile).

A.5 Neural Network Results

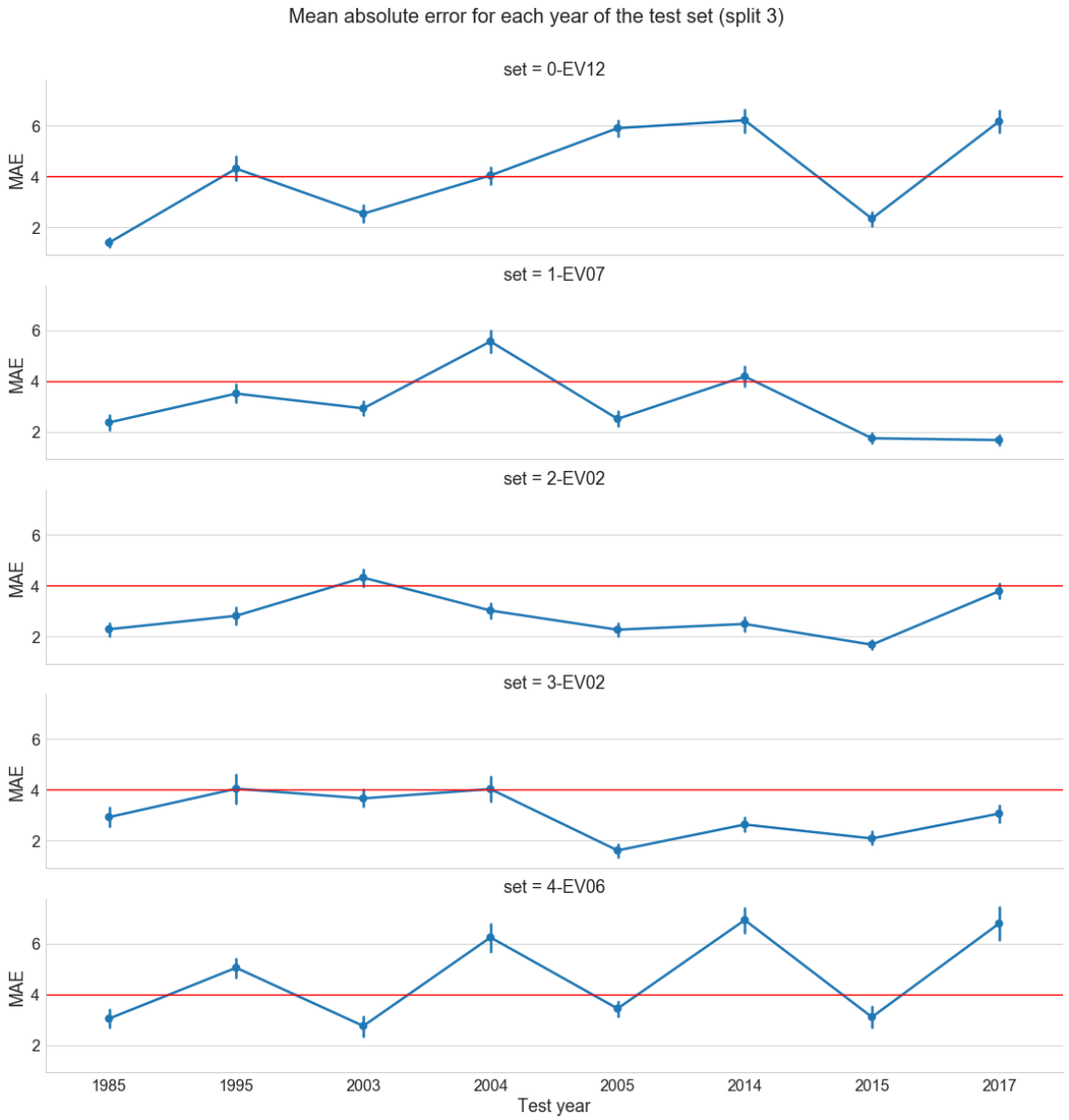

Figure A.17: Yearly mean-absolute error (with 0.95 CI) averaged over the predictions of the final experiments on the test set (1985, 1995, 2003, 2004, 2005, 2014, 2015, 2017).

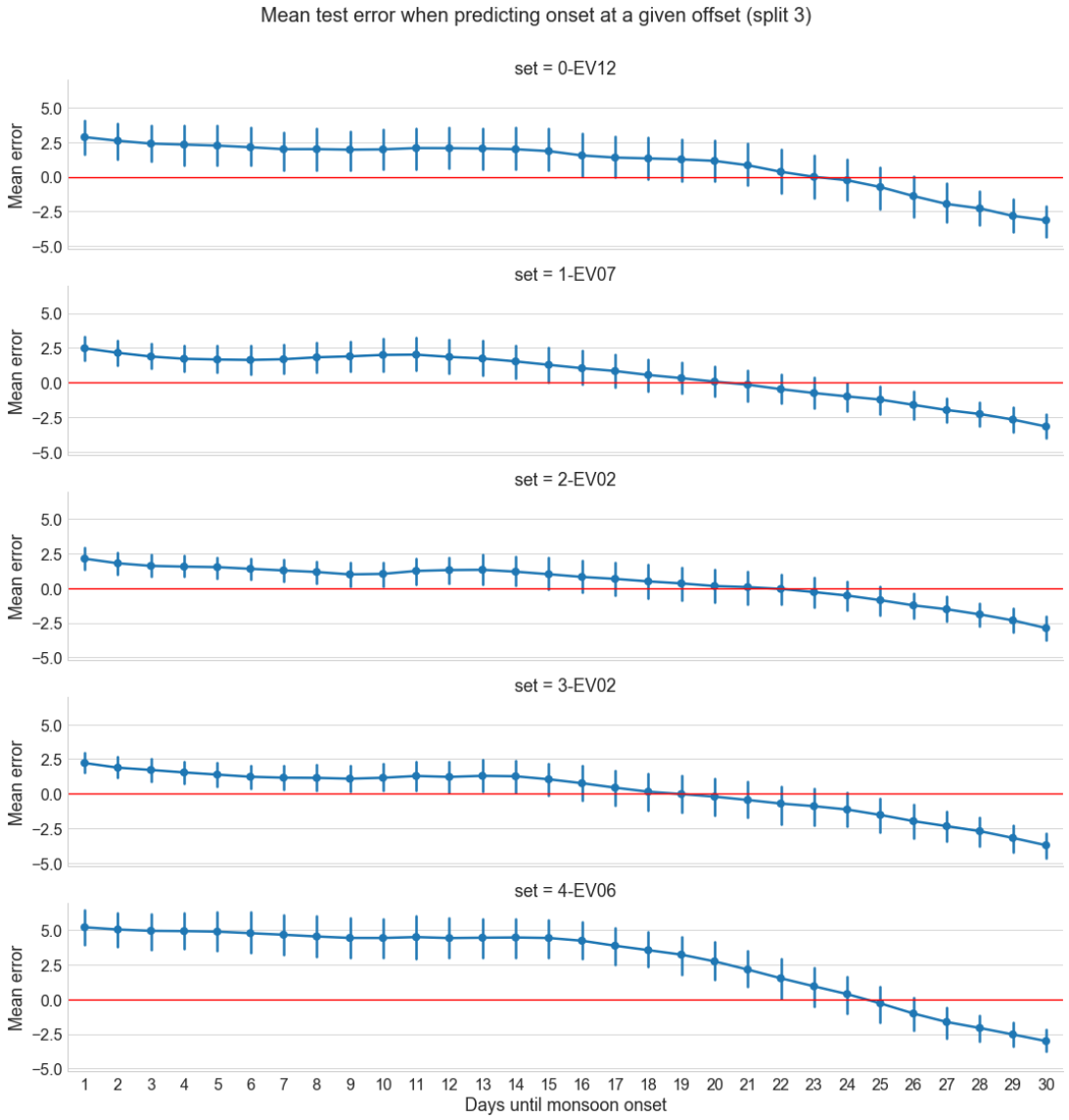

Figure A.18: Mean error (with 0.95 CI) of each offset from MoK, averaged over the predictions of the final experiments on the test set (1985, 1995, 2003, 2004, 2005, 2014, 2015, 2017).

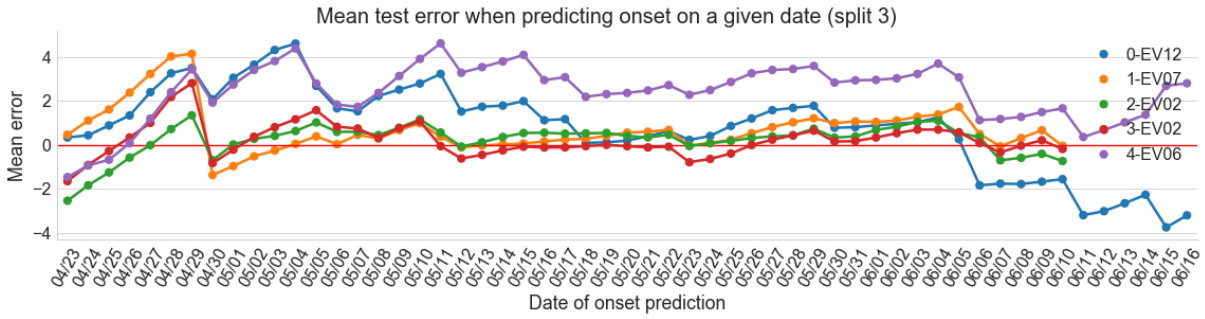

Figure A.19: Mean error (with 0.95 CI) of each prediction date previous to MoK, averaged over the predictions of the final experiments on the test set (1985, 1995, 2003, 2004, 2005, 2014, 2015, 2017).

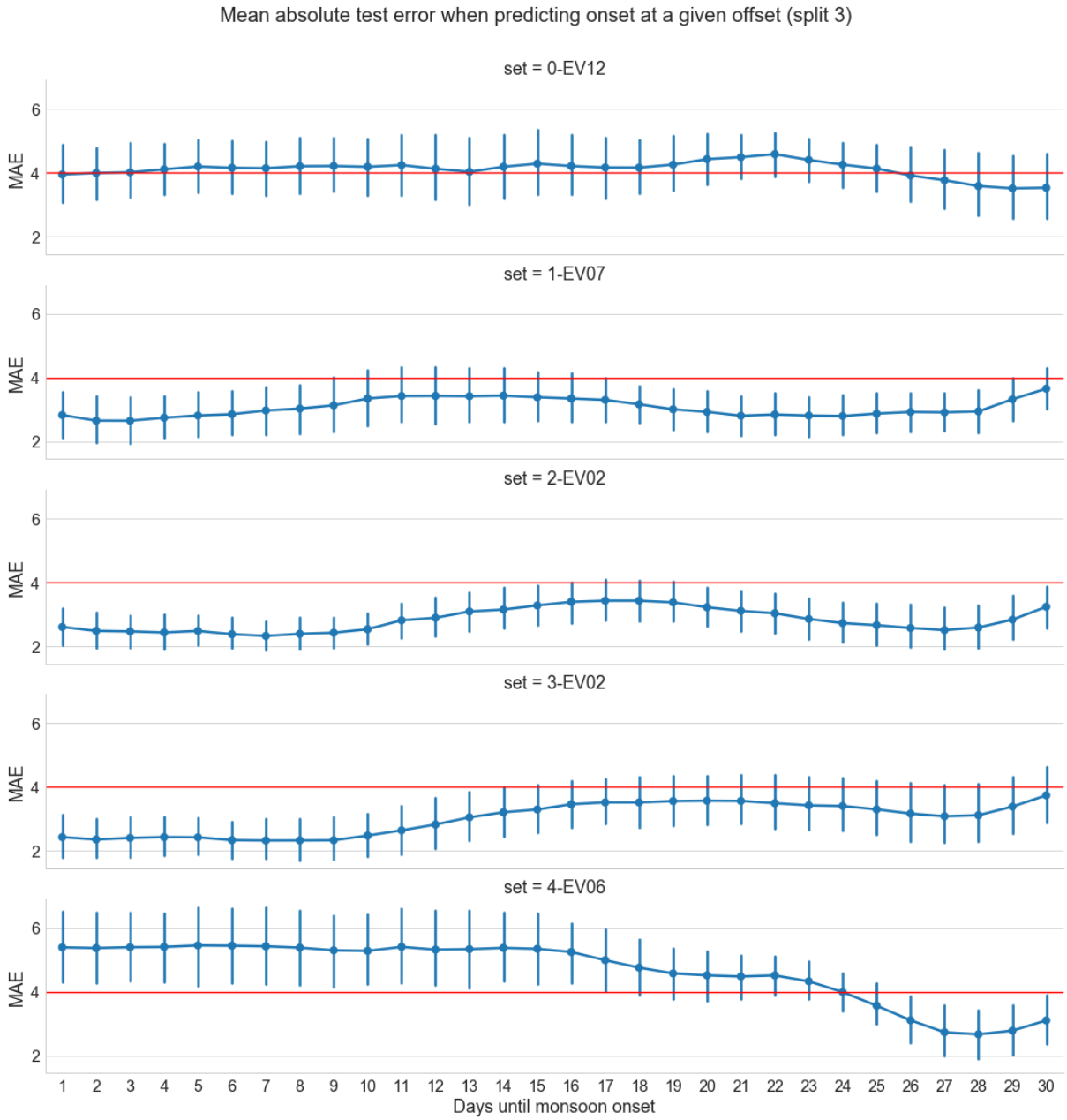

Figure A.20: Mean-absolute error (with 0.95 CI) of each offset from MoK, averaged over the predictions of the final experiments on the test set (1985, 1995, 2003, 2004, 2005, 2014, 2015, 2017).

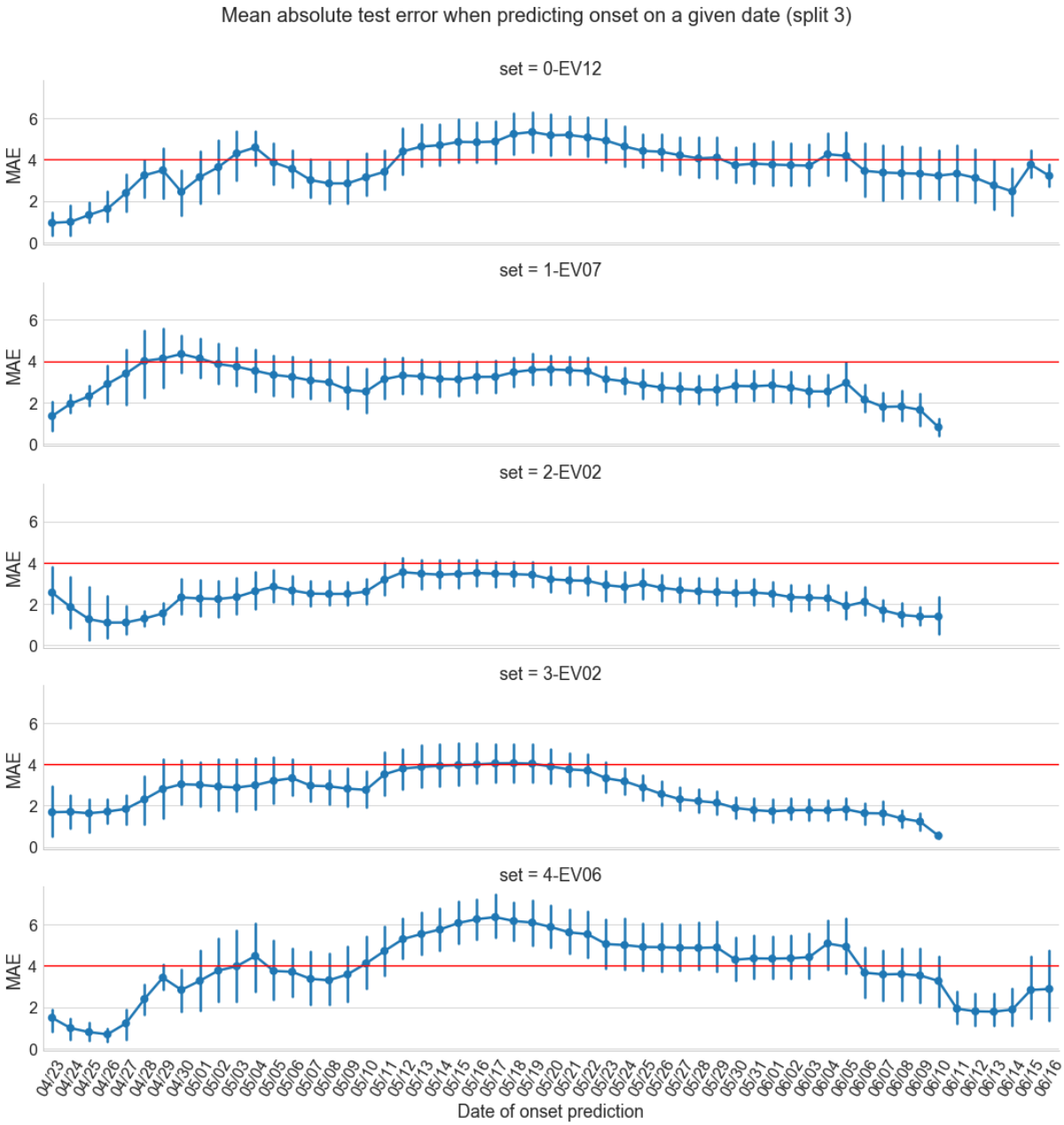

Figure A.21: Mean-absolute error (with 0.95 CI) of each prediction date previous to MoK, averaged over the predictions of the final experiments on the test set (1985, 1995, 2003, 2004, 2005, 2014, 2015, 2017).

List of Figures

List of Tables

List of source codes