# Characterizing Extreme Weather in a Changing Climate

by

Prabhat

A dissertation submitted in partial satisfaction of the

requirements for the degree of

Doctor of Philosophy

in

Earth and Planetary Sciences

in the

Graduate Division

of the

University of California, Berkeley

Committee in charge:

Professor William D. Collins, Chair
Professor David Romps
Professor Bruno Olshausen
Dr. Michael Wehner

Spring 2020

Characterizing Extreme Weather in a Changing Climate

Abstract

# Characterizing Extreme Weather in a Changing Climate

by

Prabhat

Doctor of Philosophy in Earth and Planetary Sciences

University of California, Berkeley

Professor William D. Collins, Chair

Climate change is arguably one of the most pressing challenges facing humanity in the 21st century. Over the past three decades, substantial progress has been made in characterizing the changes in earth's mean climate state under a range of global warming scenarios. However, individuals and governments across the world are now interested in *changes to the extremes* of the climate system. The tails of the distribution, typically associated with extreme weather patterns such as tropical cyclones, atmospheric rivers and extra-tropical cyclones are highly impactful, and cause major disruption to human life, property and economy. This thesis makes an attempt to address the following question: *How will extreme weather events change in the future?*

In order to address this question, we first need to address how extreme weather events are identified in climate datasets. For over four decades, climate analysts have applied heuristics on spatial and temporal summaries of complex datasets to identify events. This thesis makes the following major advances in bringing the field of climate analytics to the 21st century: First, we develop the TECA (Toolkit for Extreme Climate Analytics) framework and enable heuristic-based analysis of O(10) TB datasets in 10s of minutes. Second, we introduce Deep Learning to the climate science community, and showcase state-of-the-art results in pattern classification, detection and segmentation. Our Deep Learning implementations have been scaled to the largest CPU- and GPU-based HPC systems in the world; these achievements being recognized by the ACM Gordon Bell Prize in 2018.

Powered by these new capabilities, we characterize changes in the frequency and intensity of tropical cyclones, atmospheric rivers and extra-tropical cyclones in a variety of historical, reanalysis and climate change runs. Deep Learning-powered segmentation provides us with the capability to conduct precision analytics of precipitation *conditional* on these weather patterns. We report on thermodynamic and dynamic mechanisms behind changes in extreme precipitation.

A Word of Thanks.

I am indebted to members of my PhD thesis committee: Prof. William D. Collins, Prof. David Romps, Prof. Bruno Olshausen and Dr. Michael Wehner. I couldn't have asked for more from a thesis advisor in Bill: he has given me space to explore ideas and carry them through from conception to execution. His patience, professional demeanour and integrity are all qualities I aspire to imbibe over the course of my career. David and Bruno are legends in their own right; I feel privileged to have had the opportunity to interact with them during my time at UC Berkeley. Michael has always been the perfect colleague: encouraging new ideas, providing hands-on help with datasets, analysis, critique, edits and getting papers out of the door! I am thankful to Bruce Buffett and Jeff Chambers for serving on the Qualifying Examination Committee.

The EPS community at UC Berkeley is a remarkable mix of grounded intellectuals. Inez Fung, John Chiang and Michael Manga have been a pleasure to interact with. I was fortunate to participate and interact with members of climate reading group (Sol Kim, Jesse Day, Percy Link, Jiabin Liu) and the Romps research group (Nadir Jeevanjee, Jake Seeley and Jake Edman). Ben Fildier was always available to answer questions related to extreme precipitation. Thanks to Margie Winn for helpful pointers throughout the course of the PhD and Britney Stewart for assistance with wrapping up the thesis filing process during the unprecedented covid-19 outbreak.

Attempting to complete a PhD while being a full-time employee turned out to be quite the undertaking. This simply would not have been possible without the support of my managers, peers and staff members at Berkeley Lab. I started this journey as a Research Scientist in the Visualization Group at CRD; Wes Bethel and David Brown were extremely supportive in my efforts to start my PhD program. Life took a turn when the opportunity to transition to a group lead opened up at NERSC; Katie Antypas and Sudip Dosanjh were equally supportive in my continuation of the PhD degree. Over the course of the past 6 years, I was instrumental in creating the Data and Analytics Group at NERSC; I want to thank all group members for their patience and forbearance. Karthik Kashinath, Thorsten Kurth, Mayur Mudigonda, Evan Racah and Wahid Bhimji in particular have been deeply engaged in a number of projects conceived and executed over the course of this thesis. Berkeley Lab has funded my PhD through its generous Tuition Assistance Program; I'd like to call out Emalynn Robinson for ensuring that all of my paperwork was completed on time! It was a pleasure to engage with members of the CASCADE team (Travis O'Brian, Chris Paciorek, Burlen Loring, Suren Byna, Soyoung Jeon, Ankur Mahesh) over the course of my studies.

One of the unique privileges of my role at NERSC was the opportunity to engage a variety of collaborators from the the industry and DOE. An exhaustive list of collaborators is mentioned in the subsequent chapters; I'd like to call out Nadathur Satish, Narayanan Sundaram, Mostofa Patwary, Pradeep Dubey (Intel), Sean Treichler, Josh Romero, Mike Houston (nvidia), Eli Dart (ESnet), Arjun Shankar (OLCF) and Venkat Vishwanath (ALCF) for their critical input. Both the 'Deep Learning @ 15PF' and 'Exascale Deep Learing' projects were the highlights of my professional career at NERSC, thanks in a large part to these high quality collaborators.

I'd like to thank Jean Joseph for his assistance with proofreading and collating this document. I'd also like to thank my co-authors for allowing me to include previously published content as a part of the thesis narrative.

Finally, and most importantly, any extended endeavor requires the full support of family members. My parents Balbir and Santosh deserve all the credit for raising and educating me in India, and supporting my career in the US. Thanks to my younger brother Deepak for his advice and support over the years. I want to thank Olga for bringing Kartik into this world; his presence has brought balance and serenity to my life. My life is uniquely blessed to be associated with three souls: the perfect self-enlightened being, the being who has attained enlightenment through effort, and the being who was born destined for perfection. Upon reflection, I can only submit the following verse from an anonymous Urdu poet:

*That exalted status is yours,*
*When I bow my head before you;*
*Yet this head I bow*
*Is but a gift from you to me.*
*Far more than my destiny allows*
*From your gracious hands I receive;*
*Yet even this destiny*
*Is but a gift from you to me.*

# Contents

# List of Figures

# List of Tables

# Acknowledgments

## Chapter 4

**Pattern Recognition with TECA:** This work was supported by the Regional and Global Climate Modeling Program and the Earth System Modeling Program of the Office of Biological and Environmental Research, the Director, Office of Science, Office and Advanced Scientific Computing Research, of the U.S. Department of Energy under Contract No. DE-AC02-05CH11231. This work used resources of the National Energy Research Scientific Computing Center (NERSC), which is supported by the Office of Science of the U.S. Department of Energy under Contract No. DE-AC02-05CH11231.

**Deep Learning for Classification:** This research was conducted using "Neon", an open source library for deep learning from Nervana Systems. This research used resources of the National Energy Research Scientific Computing Center, a DOE Office of Science User Facility supported by the Office of Science of the U.S. Department of Energy under Contract No. DE-AC02-05CH11231. This work was supported by the Director, Office of Science, Office of Advanced Scientific Computing Research, Applied Mathematics program of the U.S. Department of Energy under Contract No. DE-AC02-05CH11231.

**Deep Learning for Detection:** This research used resources of the National Energy Research Scientific Computing Center (NERSC), a DOE Office of Science User Facility supported by the Office of Science of the U.S. Department of Energy under Contract No. DE-AC02-05CH11231. Code relies on open-source deep learning frameworks Theano (Bergstra et al.; Team et al., 2016) and Lasagne (Team, 2016), whose developers we gratefully acknowledge. We thank Samsung and Google for support that helped make this research possible. We would also like to thank Yunjie Liu and Michael Wehner for providing access to the climate datasets; Alex Lamb and Thorsten Kurth for helpful discussions.

**Spatial Statistics:** This research was supported by the Director, Office of Science, Office of Advanced Scientific Computing Research, of the U.S. Department of Energy under Contract No. DE-AC02-05CH11231. This work was funded under the CASCADE Science Focus Area funded by DOE/BER as part of their Regional and Global Climate Modeling Program. This research used resources of the National Energy Research Scientific Computing Center.

## Chapter 5

**Scaling Deep Learning on CPUs:** This research used resources of the National Energy Research Scientific Computing Center (NERSC), a DOE Office of Science User Facility supported by the Office of Science of the U.S. Department of Energy under Contract No. DE-AC02-05CH11231. We would like to thank Doug Jacobsen, Brandon Cook, Tina Declerck, David Paul and Rebecca Hartman-Baker for assisting with and troubleshooting Cori reservations. We would like to acknowledge Christopher Beckham, Tegan Maharaj and Christopher Pal, Yunjie Liu and Michael Wehner for help with preparing the climate architecture and dataset. Christopher Ré's group at Stanford was the source of valuable advice on asynchrony. We would like to thank Srinivas Sridharan, Mikhail Smorkalov, Mikhail Shiryaev and Dipankar Das for their help in integrating and modifying Intel MLSL.

**Exascale Deep Learning on GPUs:** This research used resources of the National Energy Research Scientific Computing Center (NERSC), a DOE Office of Science User Facility supported by the Office of Science of the U.S. Department of Energy under Contract No. DE-AC02-05CH11231. This work was supported by a grant from the Swiss National Supercomputing Centre (CSCS) under Project ID g107. We thank Nicholas Cardo, Andreas Joksch, Miguel Gila and the CSCS staff for assistance in using Piz Daint. We thank Paul Tucker and Rajat Monga from Google for helpful discussions pertaining to TensorFlow. Michael Wehner, Karthik Kashinath, Burlen Loring, Travis O'Brien and Bill Collins from LBNL were instrumental in motivating the climate science problem and providing datasets. This research used the Summit system at the Oak Ridge Leadership Computing Facility at the Oak Ridge National Laboratory, which is supported by the Office of Science of the U.S. Department of Energy under Contract No. DE-AC05-00OR22725. We are very grateful to OLCF staff: Veronica Melesse Vergara; Don Maxwell, and Matthew Ezell for their assistance with the runs, and Arjun Shankar; Ashley Barker; Tjerk Straatsma and Jack Wells for programmatic support.

# Chapter 1

# Extreme Weather and Climate Change

## 1.1 Extreme Weather Events and their Impact

Climate change is arguably one of the most pressing challenges facing humanity in the 21st century. There are fundamental uncertainties in climate projections stemming from the approximations made by computational models, uncertainty in the observations, assumptions regarding carbon emission and intervention strategies from governments. Nevertheless, substantial progress has been made in characterizing the changes in the mean state (temperature, precipitation, etc) of the system [92].

Increasingly, individuals around the world, local/state and federal governments are interested in better understanding *changes to the extremes* of the climate system, in addition to its mean properties. Arguably, tails of the distribution, in terms of extreme weather events, are highly impactful and cause major disruption to human life, property and the broader local/global economy. Billion dollar weather-related disasters are increasing in frequency [161]; extreme weather events caused over $32B of property damage in 2017 [85, 161]; wind and tornadoes have killed an average of 160 people annually in the last 10 years [150]; and it is estimated that weather and climate variability accounted for $3.86 trillion of the US Gross Domestic Product (GDP) in 2000 [43].

Work conducted under this thesis tries to make progress in addressing the following simple question: *How will extreme weather events change in the future?*

The answer to this important question can inform a range of stakeholders: Policy makers at the state level may choose to implement policies regarding water resource management, Decision makers at the city level may choose to prioritize funding for building levees, reinsurance companies may choose to evaluate pricing structure for their offerings, international firms may choose to mitigate risk in their supply chains, individuals may choose to buy housing in a different region, and so on.

There are a variety of extreme weather events (heat waves, tornadoes, hail, mesoscale convective complexes, etc) that can impact human life; we will now enumerate a few important classes of events that we have chosen to study in this thesis:

### 1.1.1 Tropical Cyclones

Figure 1.1: Hurricane Katrina (left) and its impact on the city of New Orleans, Louisiana (right).

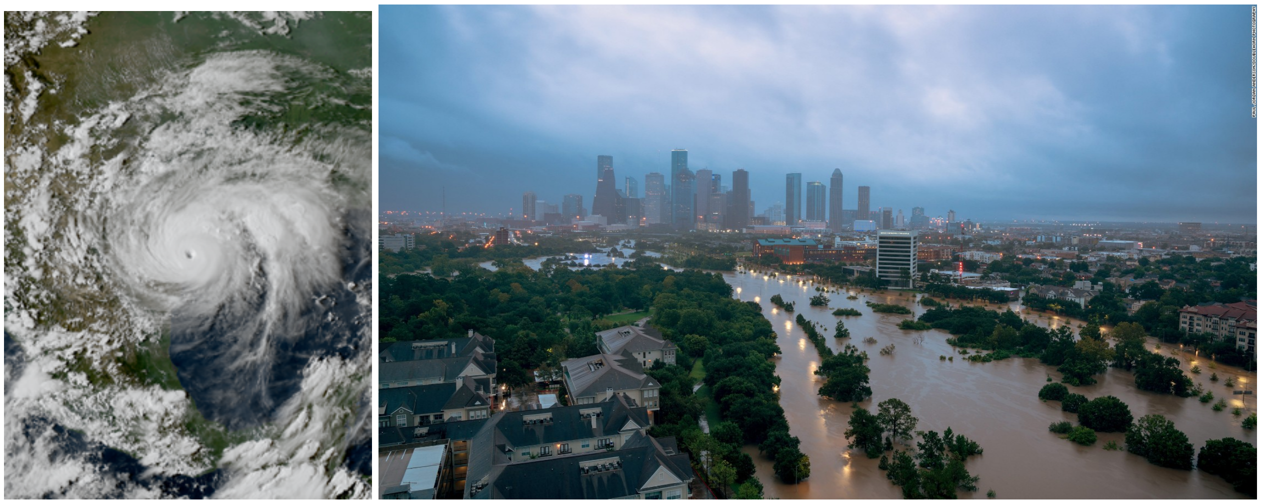


Figure 1.2: Hurricane Harvey (left) and its impact on the city of Houston, Texas (right).

Tropical storms are one of the most damaging climate events in terms of monetary destruction and loss of human life [262]. Hurricanes Katrina and Harvey (depicted in Figures 1.1 and 1.2 are contemporary examples of the devastation caused by events that made landfall on major US cities. Better understanding of the future trend of frequency and severity of the tropical storms is critical to assessing impacts of climate change. High-resolution climate models, run under different scenarios of human activity, provide a tool to project these trends [262, 260, 109]. These models are tested by performing simulations of the recent past, so their output can be compared against available observations. After assessing a model against these observations, they are run far into the future to investigate how tropical storm characteristics might change.

### 1.1.2 Atmospheric Rivers

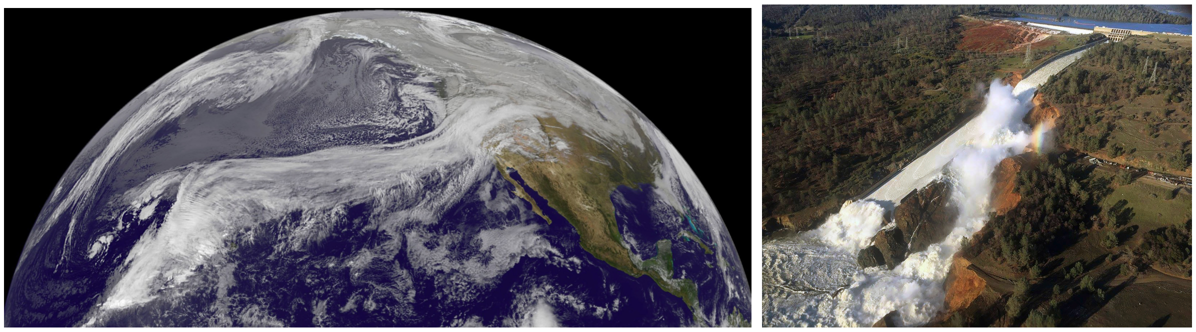


Figure 1.3: Atmospheric River event (left). Impact of consecutive atmospheric river events on the Oroville Dam in California.

An atmospheric river is a long and narrow structure in atmosphere that transports tropical moisture to the far-flung regions outside of the tropical zone [39, 199]. Zhu and Newell were the first to name this phenomenon "atmospheric river" noting that they typically transport more water than the Amazon[283]. As they can be highly localized, "river" is an apt description of such a narrow stream of moisture moving at high speeds across thousands of kilometers. After transporting moisture, atmospheric rivers may run into orographic features, resulting in uplift, and subsequent precipitation. The state of California receives 50% of its annual precipitation in the winter through this important mechanism. Understanding of how atmospheric rivers, and their associated precipitation characteristics, might change in future climate regimes is an important problem. Should the frequency of atmospheric rivers decrease in future climate regimes, state government agencies responsible for water resource planning will need to make alternate plans. Conversely, should atmospheric rivers become more frequent, they may result in catastrophic flooding (depicted in Figure 1.3).

### 1.1.3 Extra-Tropical Cyclones

Extra-tropical cyclones are of key importance to the climate at middle to high-latitude regions. They transport heat, momentum and moisture, determine the local weather systems and also have strong influence on the general circulation [9, 119, 118]. For instance, the traveling directions of Atlantic cyclones have been shown to determine the precipitation and temperature in northwestern and southern Europe [131]. Extra-tropical cyclones are characterized by lower winds and less intense pressure minima than their tropical counterparts, nevertheless they often result in heavy gales, thunderstorms, and other severe weather. They

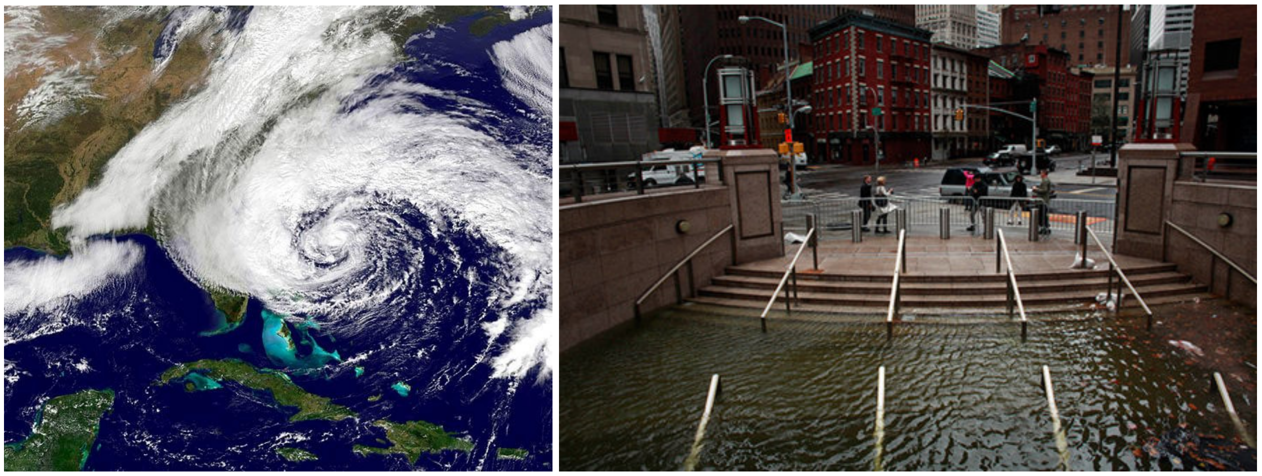


Figure 1.4: Extra-Tropical Cyclone Sandy (left) and its impact on the Manhattan, New York (right).

are also generally much larger in their geographical extent than tropical storms and occur more frequently. Hurricane Sandy (depicted in Figure 1.4) is a contemporary example of an extra-tropical cyclone that made landfall on a major metropolitan area. A good understanding of how the statistics of extra-tropical cyclones (intensity, frequency, or position) change with the changing climate will be fundamental for the projection of the local climate in the middle to high-latitude regions.

## 1.2 Current Analysis Techniques and their Limitations

Detection and Analysis of extreme weather patterns *requires* exploration of the highest fidelity datasets (both in terms of spatial and temporal resolution). This is in marked contrast to typical analysis in climate science which deals with global/zonal/local spatial aggregations, and annual/seasonal/daily summary quantities. While high resolution datasets now exist in climate science, the state-of-the-art in analysis methods has simply not kept pace with the requirements to analyze extreme weather events. In this thesis, we make substantial progress along two lines of investigation: proposing modern, machine learning methods instead of decade-old heuristics, and scaling these methods on large HPC systems.

### 1.2.1 Advancing the state-of-the-art beyond heuristics

Identifying extreme weather events is a crucial first step in understanding how they may vary under different climate change scenarios. To do so, climate scientists have largely relied on custom *heuristics* for the identification of these events [79, 157, 183, 221, 244]. However, there are often large discrepancies between different detection algorithms for the same type

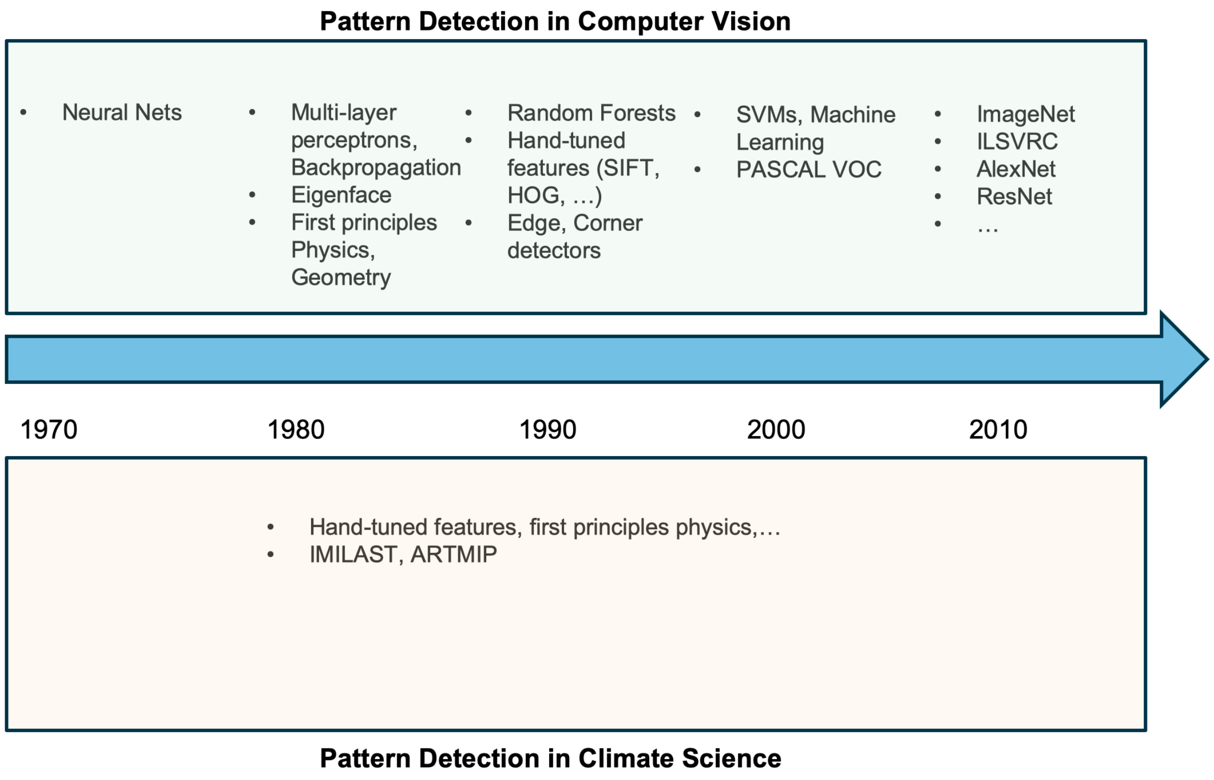

Pattern Detection in Computer Vision
Neural Nets
Multi-layer perceptrons, Backpropagation
Eigenface
First principles Physics, Geometry
Random Forests
Hand-tuned features (SIFT, HOG, …)
Edge, Corner detectors
SVMs, Machine Learning
PASCAL VOC
ImageNet
ILSVRC
AlexNet
ResNet
…
1970
1980
1990
2000
2010
Hand-tuned features, first principles physics,…
IMILAST, ARTMIP
Pattern Detection in Climate Science

Figure 1.5: Comparison of techniques employed in computer vision community versus climate science community

of pattern or event. Different heuristics rely on different subsets of variables, and choices of threshold conditions. Often there are large discrepancies on the overall count of such events, and their spatial extents.

As an illustration of this limitation, many different atmospheric river (AR) detection algorithms exist that produce largely different outputs. This recently motivated researchers to launch the Atmospheric River Tracking Methods Intercomparison Project (ARTMIP), which found that AR counts can differ by an order of magnitude depending on which algorithm is used [222]. Similarly, the IMILAST [157] project sought to compare detection algorithms for Extra-Tropical Cyclones (ETC); and concluded that various ETC detection methods produce up to widely varying estimates (3-7x) for ETC counts. A related (but understudied) issue pertains to heuristics for defining the *spatial extent* of weather patterns [19, 3, 51, 110, 176] . Given the wide discrepancy in the storm counts, we have limited reason to believe that the heuristics pertaining to storm extents fare any better. It is noteworthy that these issues have plagued the climate analytics/informatics community over the past 30 years, and it is unclear as to what the solution might be (i.e. development of yet more heuristics, weighted combinations of heuristic output, Bayesian or probabilistic treatment of heuristic output, etc.). In this thesis, we will make fundamental contributions in applying Deep Learning techniques towards solving this open problem.

### 1.2.2 Enabling analysis of massive datasets

Modern climate simulations produce massive amounts of data. Commonly used models such as the Community Earth System Model (CESM) and Weather Research and Forecasting model (WRF) routinely produce TBs of data. Next generation Global cloud resolving simulations will produce TBs of data on a per-timestep basis. Coordinated multi-model ensembles of climate model simulations such as the Coupled Model Intercomparison Project (CMIP3) [143] contain hundreds of GBs; CMIP5 [91] contains PBs of data with CMIP6 projected to host even more. Such datasets have presented tremendous opportunities in terms of sophisticated analysis (time-series, geo-spatial, pattern detection, etc.) to further our understanding of the climate system. Indeed, such ensembles are a crucial tool for international and national organizations, such as the Intergovernmental Panel on Climate Change (IPCC), tasked to assess the human role in climate change and the potential of adaptation and mitigation strategies.

While valuable insight has been gained from such datasets, the sheer size of the CMIP3 archive has presented significant challenges to the climate science community. The increased magnitude of CMIP5 in terms of both data volume as well as complexity is limiting its potential uses. Conventional visualization and analysis tools, such as CDAT [14], ferret [47], and nco [155], continue to rely on a serial execution model, fundamentally limiting the applicability of such tools to large datasets. As higher resolution climate models enter the mainstream, insurmountable obstacles will face the climate community if these limitations are not removed. While efforts, such as UV-CDAT [245], ParCAL [173] and PAGODA [171] are currently underway, advanced capabilities, such as machine learning and spatio-temporal analysis, remain out of reach for most tools. Such limitations on the ability of climate scientists to exploit these large datasets are unnecessary, as large supercomputing systems and commodity clusters are now commonly available and provide ample horsepower to facilitate sophisticated analysis. In this work, we build on current parallel computing hardware and distributed memory programming models to develop capabilities for analyzing modern massive climate datasets.

Climate model output exposes ample parallelism across multiple dimensions. Typically, it is possible to run the analysis code across individual ensemble members, timesteps, spatial regions and individual grid points. For instance, hurricane detection can be performed independently across models with perturbed physics (ensembles), years, spatial regions (northern hemisphere vs. southern hemisphere), and grid locations. Therefore, it is imperative that we design and develop analysis capabilities that exploit the high degree of parallelism in climate datasets and overcome the large data challenges in the climate community. In this thesis, we will develop a general Toolkit for Extreme Climate Analytics (TECA) framework to enable analysis codes to run at unprecedented scale on HPC systems.

## 1.3 Contributions of this Thesis

This thesis has made the following contributions:

- **Development of State-of-the-Art Analytics Methods**: Chapter 4 presents a range of advanced methods based on heuristics (TECA), spatial statistics and Deep Learning (DL). In particular, we successfully introduce Deep Learning to the climate science community, and demonstrate the first applications of DL methods for solving pattern recognition problems.

- **State-of-the-Art Scaling of Analytics Methods**: Chapter 5 showcases the scaling of our TECA and Deep Learning applications to the largest CPU- and GPU-based HPC systems in the world. We have succeeded in processing $O(10)TB$ climate datasets in 10s of minutes. These achievements have been acknowledged with the Juelich Supercomputing Center Prize, and the prestigious Gordon Bell Prize.

- **Creation of ClimateNet**: Chapter 6 highlights our efforts in creating an open, expert-labeled dataset for identifying extreme weather events. We share both the dataset, and the trained DL architecture with the broader community. Similar to the transformative impact of ImageNet on the computer vision community, we hope that the climate scientists will extend and leverage the dataset for numerous downstream applications.

- **Characterization of changes in extremes**: Chapters 2 and 3 showcase scientific results from the application of TECA, spatial statistics and Deep Learning tools to climate change scenarios. We highlights changes in storm counts and extreme precipitation, and speculate on underlying mechanisms associated with the changes.

# Chapter 2

# Changes in Frequency and Intensity of Extreme Weather Events

This thesis has developed advanced capabilities to enable analysis of extreme weather patterns. First, have developed the TECA (Toolkit for Extreme Climate Analytics) framework which enables heuristic-based approaches to scale to large datasets. Second, we have introduced Deep Learning to the climate science community, and demonstrated state-of-the-art performance in classifying, detecting and segmenting extreme weather patterns. These innovations will be discussed at length in Chapters 4, 5 and 6. We will now present the applications of these advanced methods to the problem of characterizing changes in extreme weather patterns under global warming.

## 2.1 Tropical Cyclones

One of the primary scientific utilities of the TECA software is to evaluate how well models perform in reproducing extreme event statistics, compared to observational records. If we assess models to perform well for the historical period, we can have greater confidence in the trends projected by the same models for future runs. We have applied TECA to the CAM5 0.25-degree output, over a simulated time period spanning 1979-2005 [261]. For this time period, the hand-labelled IBTrACS dataset [108] reports 87 (+/- 8) storms every year. TECA reports 84 (+/-9) storms, which is rather accurate. Figures 2.1 and 2.2 highlight the spatial distribution of the storms, as well as the seasonal distribution. In terms of model evaluation, we note that CAM5 does a good job of reproducing the spatial pattern. Although the model produces the correct number of tropical storms in the Northern Pacific basin, there are too many storms in the central Pacific, notably around Hawaii. One plausible explanation for this is an systematic high bias in the model's relative humidity in that region. Atmospheric waves that move into the region more likely develop into tropical storms in the model than they would in the real world due to the increased available latent heat energy The model also does a good job of reproducing the seasonal pattern in various ocean basins (North Atlantic,

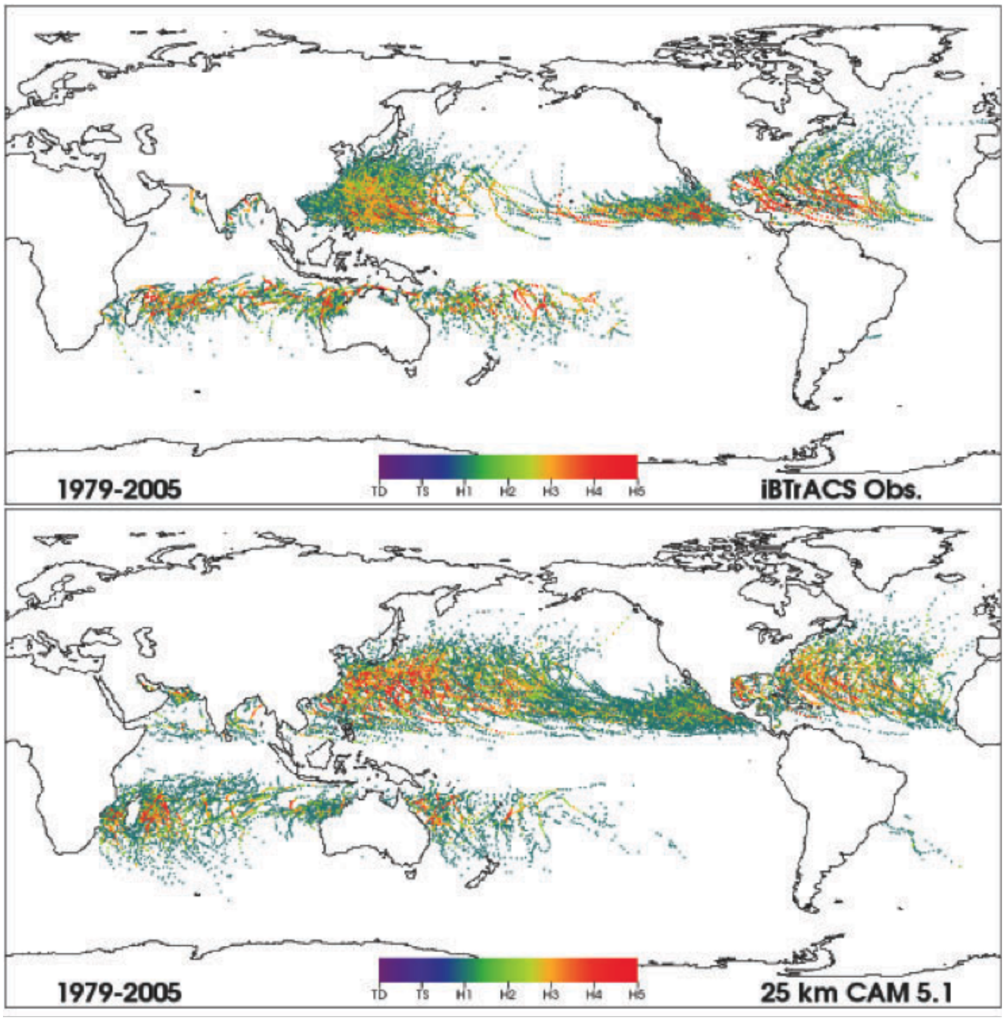


Figure 2.1: Application of TECA TC detection capability to CAM5 0.25-degree output. Tropical Cyclones (Category 1 through 5) are illustrated in the bottom figure. TC tracks from the IBTrACS observational product are plotted for an identical time period.

Indian Ocean, Northwest Pacific), but the storms counts are off in the Pacific.

After validating the TECA output for the historical period, we applied the TC detection capabilities for climate change experiments conducted by various US and international efforts. We processed a climate change experiment specified by the CliVAR Working Group [25]. We used the CAM5 model to simulated the earth's climate under a baseline (climo), a scenario consisting of 2x$CO_2$, SSTs increased uniformly by 2°C, and the conjunction of both $CO_2$ and SST conditions. Figure 2.3 shows the average number of tropical storms, tropical cyclones and intense tropical cyclones per year simulated by the high-resolution version (0.23°x0.31°) of CAM5.1 for the four idealized configurations. Error bars represent 5%-95%

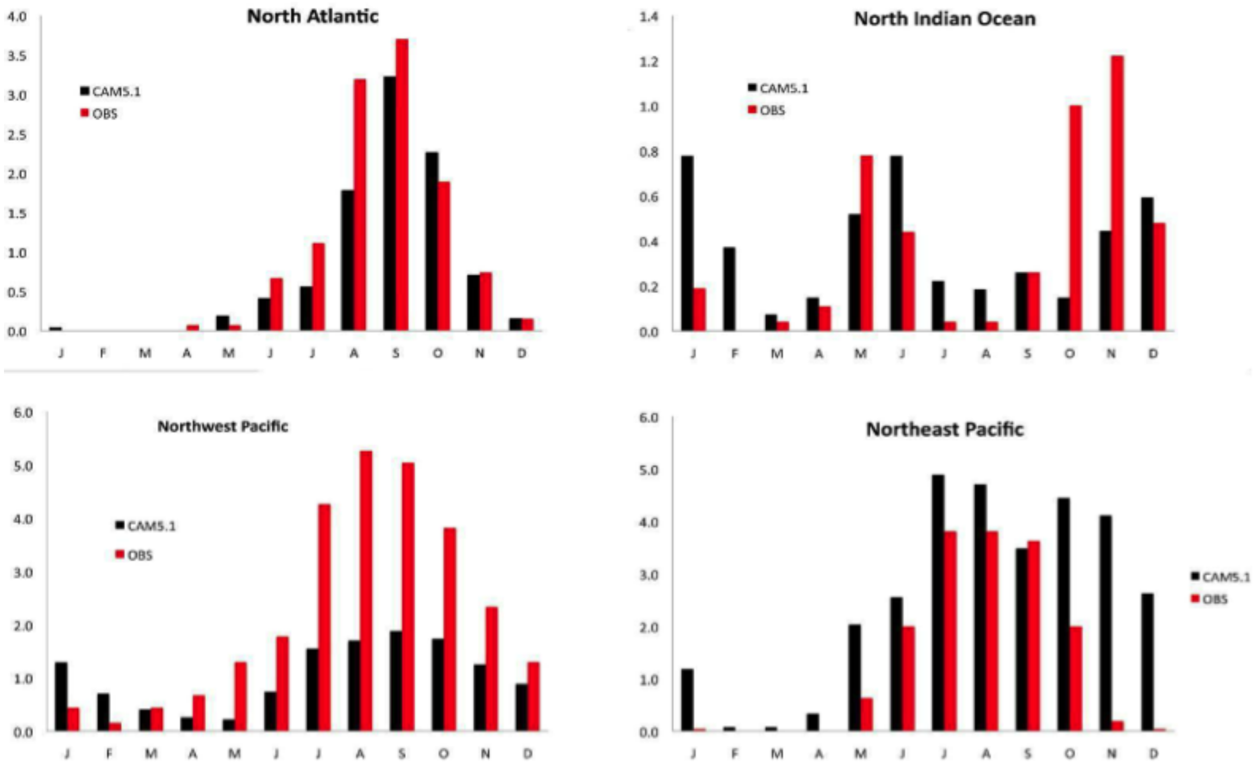


Figure 2.2: TECA can produce detailed diagnostics for storm tracks. In this case, monthly TC activity is plotted by major oceanic basins. CAM5 output is plotted in black, and observational products are plotted in red.

confidence intervals based on inter-annual variability. The baseline (1990) climatology is in blue. A two-degree warmer simulation with elevated atmospheric carbon dioxide levels (660ppm) is shown in red. Under the combined effect of the uniform SST increase and $CO_2$ doubling, the annual number of tropical storms are reduced from 86±4 to 70±3. The annual number of intense tropical cyclones (Category 4-5) increase from 10±1.7 to 12±1.7. The single-condition experiments reveal that most of the reduction in the total number of tropical storms of all intensities are caused by the change in vertical temperature profile due to $CO_2$ doubling, while the increase in the number of intense tropical cyclones is caused by the increased SST.

### 2.1.1 Mechanisms

The critical ingredients for tropical cyclogenesis involve warm sea surface temperatures (tropical oceans), presence/formation of vorticity seeds, weak wind shear and presence of high humidity in the mid-troposphere. In a warmer world, we expect more favorable conditions

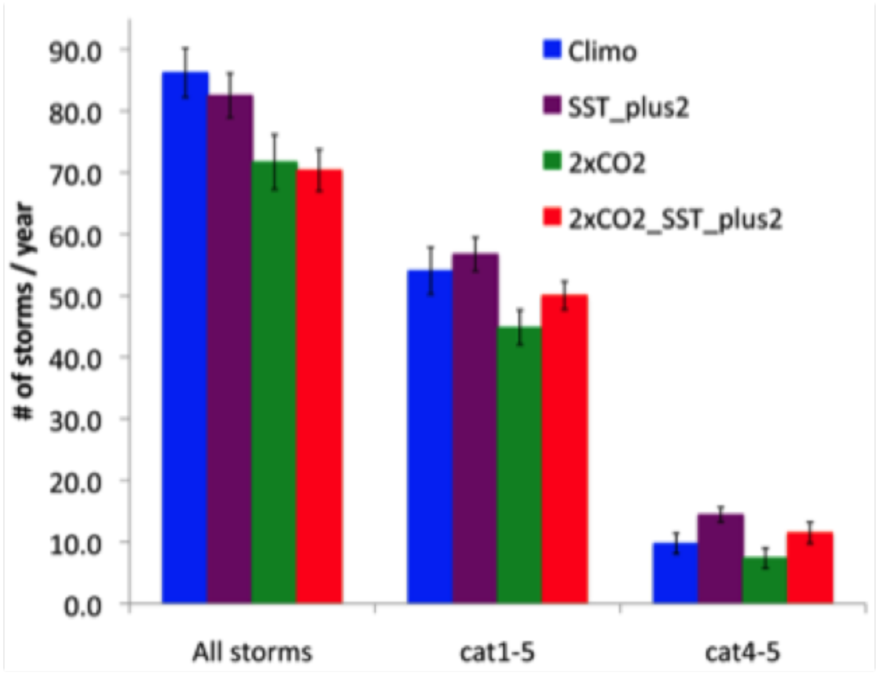


Figure 2.3: Number of annual Tropical Cyclones under the CliVAR scenarios. There is a significant decrease in the overall number of storms and an increase in number of annual Category 4 and 5 storms under the SST warming scenarios.

with regards to even warmer tropical ocean waters and higher levels of humidity. The atmosphere is expected to be vertically more stable, favoring weaker wind shear conditions, but reducing the potential for formation of vorticity seeds. Reduced number of vorticity seeds impacts cyclogenesis dramatically, affecting, in particular, the formation of weaker storms [246]. However, once cyclogenesis does happen, weaker wind shear conditions, higher levels of humidity and higher ocean surface temperature result in the successful intensification of the stronger tropical cyclones.

## 2.2 Atmospheric Rivers

We start by validating the TECA AR detection capability on the SSM/I satellite product. Figure 4.8 shows a range of diverse AR features returned by our implementation. We note that the implementation is robust to various shapes and sizes of AR events. In order to validate the procedure, we compared the events returned by TECA to a hand-curated database of known AR events maintained by [39]. We note that TECA was able to detect 93% of all events reported in the database. We furthermore applied the TECA toolkit to various CMIP-3 and CMIP-5 models over the historical period. Figure 2.4 shows that several models

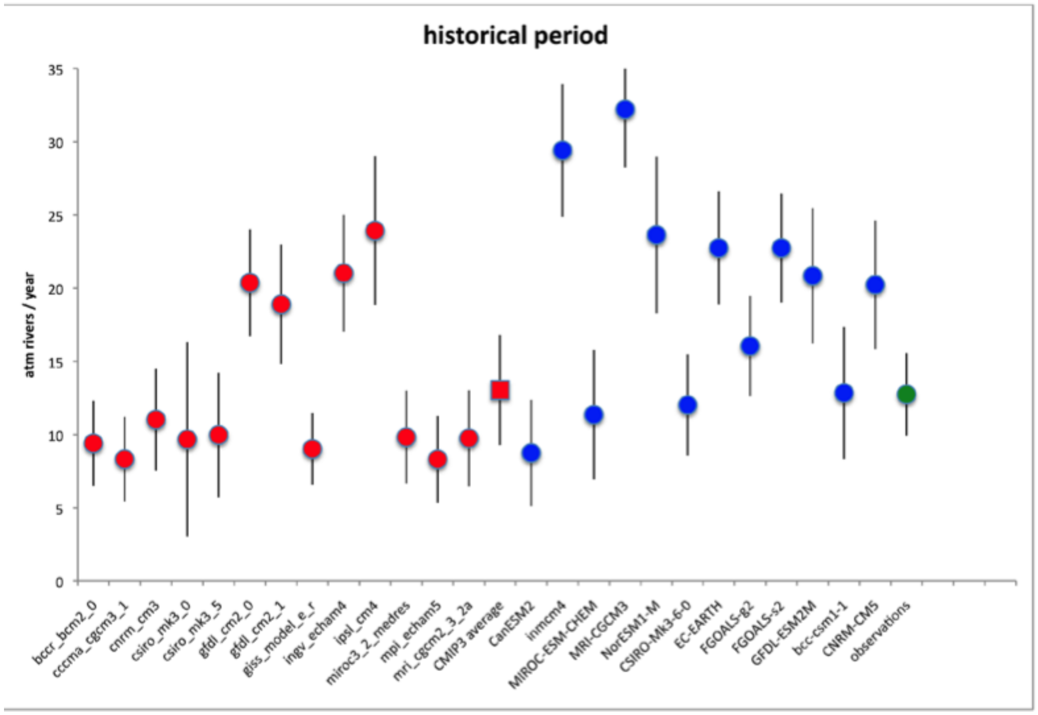


Figure 2.4: Number of Atmospheric River events in the CMIP-3 (red) and CMIP-5 (blue) archives compared to observations (green). It appears that there are two modes in the CMIP-3 and CMIP5-archive: models that are generally consistent with the observed record and models that are hyperactive in reproducing atmospheric rivers.

match reasonably well with the observed record, however, some models do exhibit hyperactivity with regards to generation of ARs. Similar to the Tropical Cyclone work, we are currently investigating the application of TECA to climate change scenarios.

We now examine changes in the overall behavior of ARs using CMIP5 multi-model ensemble simulations in a changing climate under the RCP8.5 emission scenario. When an AR satisfying the conditions—length and width of merged polygons, and *prw* exceeding a certain threshold—is detected and overlaps any portion of the California region, we treat it as an individual AR event. If the detected event lasts more than one day, it is counted as a single AR event. AR days (i.e., total days of ARs in a year) are counted for two 25-year time periods. The box plots of AR days for each model, including multi-model ensemble means of AR days, are shown for historical (blue) and RCP8.5 (red) runs in Figure 2.5. Changes in the median of the box plots for each model represent the differences in the median in a warmer climate for the model. Only IPSL-CM5A-MR shows a decrease (2 days/year) in AR days while NorESM1-M shows the largest increase (17 days/year) of AR days in a warmer scenario. Under the RCP8.5 scenario in years 2076-2100, the ensemble mean of AR days shows substantial increases—on average of 33%—compared to historical runs in years 1981-2005.

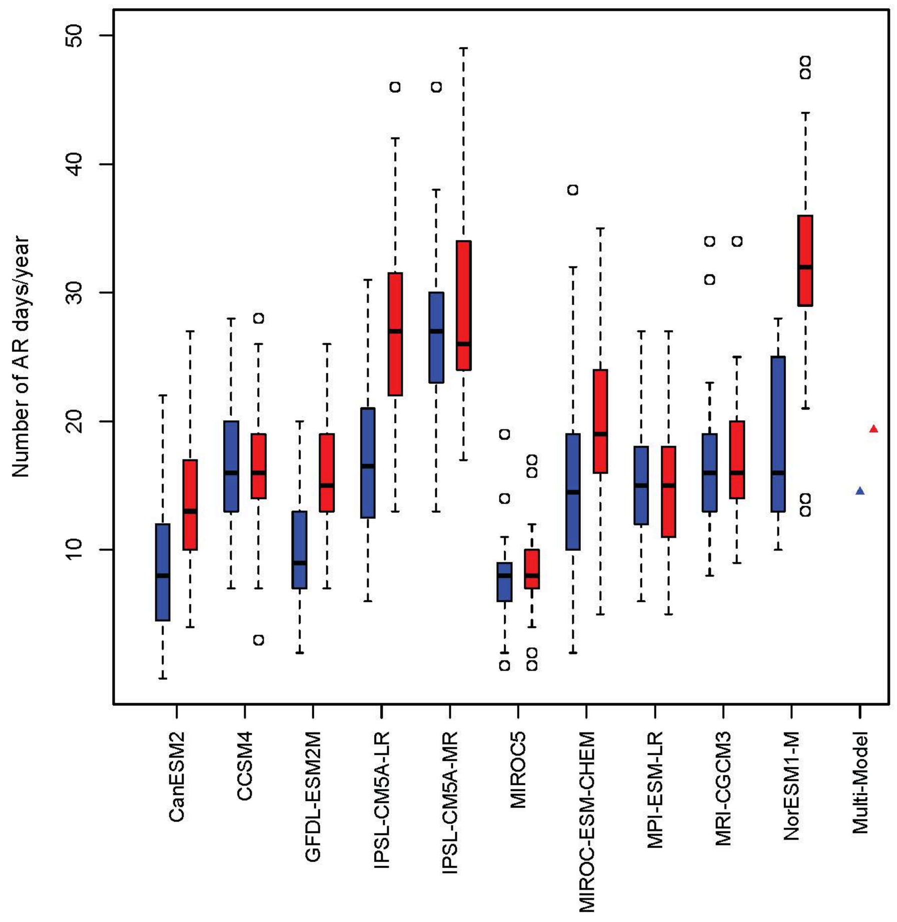


Figure 2.5: Boxplot summaries of annual values of atmospheric river days (AR days, unit: days/year) for the California region from each CMIP5 model during the 25 years, spanning 1981-2005 (blue boxplot) and 2076-2100 under RCP8.5 (red boxplot). Blue and red triangles represent the ensemble mean of AR days for the historical and RCP8.5 runs, respectively.

Figure 2.6 shows boxplots of the number of AR events in historical (blue boxplot) and RCP8.5 (red boxplot) runs for each model. AR frequencies (i.e., total AR events in a year) also increase for all models within the RCP8.5 scenario by the end of the 21st century, as indicated by median values of boxplots. The NorESM1-M simulation projects the largest change in the number of ARs with 19 events/year for the period 2076-2100 compared to 11 events/year for the period 1981-2005, representing a 73% increase in AR frequency. Most models show 2-6 more AR events per year under the RCP8.5 scenario than in the historical run.

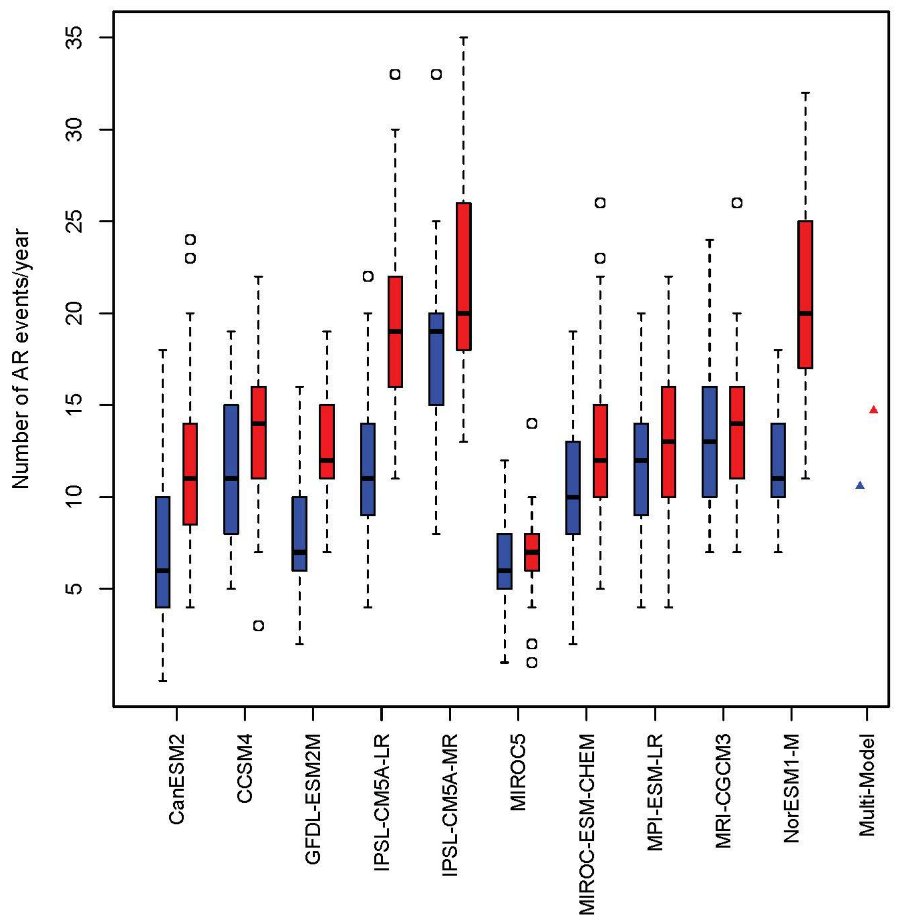


Figure 2.6: Boxplot summaries of annual numbers of atmospheric river events (AR frequency, unit: events/year) for the California region from each CMIP5 model during the 25 years, spanning 1981-2005 (blue boxplot) and 2076-2100 under RCP8.5 (red boxplot). Blue and red triangles represent the ensemble mean of AR counts for the historical and RCP8.5 runs, respectively.

Yearly statistics of ARs show ∼12-13 events between 2002 and 2010 from observations by AMSR-E (Advanced Microwave Scanning Radiometer) satellite data (Figure 6 in [13]). Ensemble mean of AR counts (10.6 events/year) for historical run matches up well with the statistics from the observational data, and the multi-model ensembles show more frequent AR events (14.7 events/year, average of 39% increase) in the future under the RCP8.5 scenario. In any case, to evaluate model performance in CMIP5 with more accurate detection of ARs, more comparisons of climate simulations and observational datasets need to be performed for consistent time periods. As shown in Figures 2.5 and 2.6, AR events, as counted by TECA, are consistently longer and more frequent under the RCP8.5 scenario (2076-2100) than in the historical run (1981-2005) due mainly to the increased oceanic specific humidity in the warmer climate. Lavers et al. [121] also showed that AR frequency in the Northern Atlantic is approximately doubled under the RCP8.5 scenario for the period 2074-2099.

### 2.2.1 Mechanisms

One of the issues plaguing our current AR detection procedure based on absolute thresholds (e.g. $prw > 2cm$) is that when water vapor (in a warmer world) increases, grid cells that were just below the 2cm threshold suddenly become valid candidates for being considered as Atmospheric Rivers. Hence to some extent, the increase in the total number of AR events can be attributed to the limitations of thresholding based techniques. We note that we have made contributions to developing and applying threshold-free, topology-based techniques for addressing these limitations [148]. In Chapter6, we will comment on the use of Deep Learning for avoiding absolute thresholds.

A robust trend in the changes in atmospheric rivers in future climate regimes is the increase in precipitation intensity. Sections 3.2 and 3.2.3 will quantify the increase, and attribute it to predominantly thermodynamic (Clausius-Clapeyron) scaling. Other authors [51] and [178] have conducted contemporaneous analysis of changes in CMIP-5 atmospheric rivers (that make landfall on the west coast of North America) in response to RCP8.5. They report increase in moisture flux with the thermodynamic effects being most dominant in areas of peak AR frequency. They also report decreases in zonal wind baselines, weakening of 850 hPa zonal wind speed in the jet exit region, and a broadening of the jet maxima compared to historical mean positions. The dynamical effects are secondary, and prominent equatorward of the peak AR frequency regions.

## 2.3 Extra-Tropical Cyclones

We have successfully applied TECA to detect Extra-Tropical Cyclones in climate data. In perhaps the leading example of Scientific Big Data analytics in climate science, we scaled TECA to process the entire CMIP-5 archive (historical and RCP8.5 runs, all ensemble members, 6-hourly data) in one shot on 755,200 cores of the Mira IBM BG/Q system. the processing was completed in 1.5 hours. Figure 2.7 illustrates the results obtained from the TECA analysis. A few observations are noteworthy: even for the same algorithm, there is a broad spread in the absolute number of extra-tropical cyclones across various models, as well as reanalysis products. The annual count varies from $\approx$ 1800-4000 storms. This variability has been reported by other researchers [157] and highlights a fundamental limitation of heuristic based approaches for ETC detection. Nevertheless, we do see evidence for a robust *decrease* in the ETC counts across all CMIP-5 models; this agreement in the sign of the change across a large multi-model archive is rare.

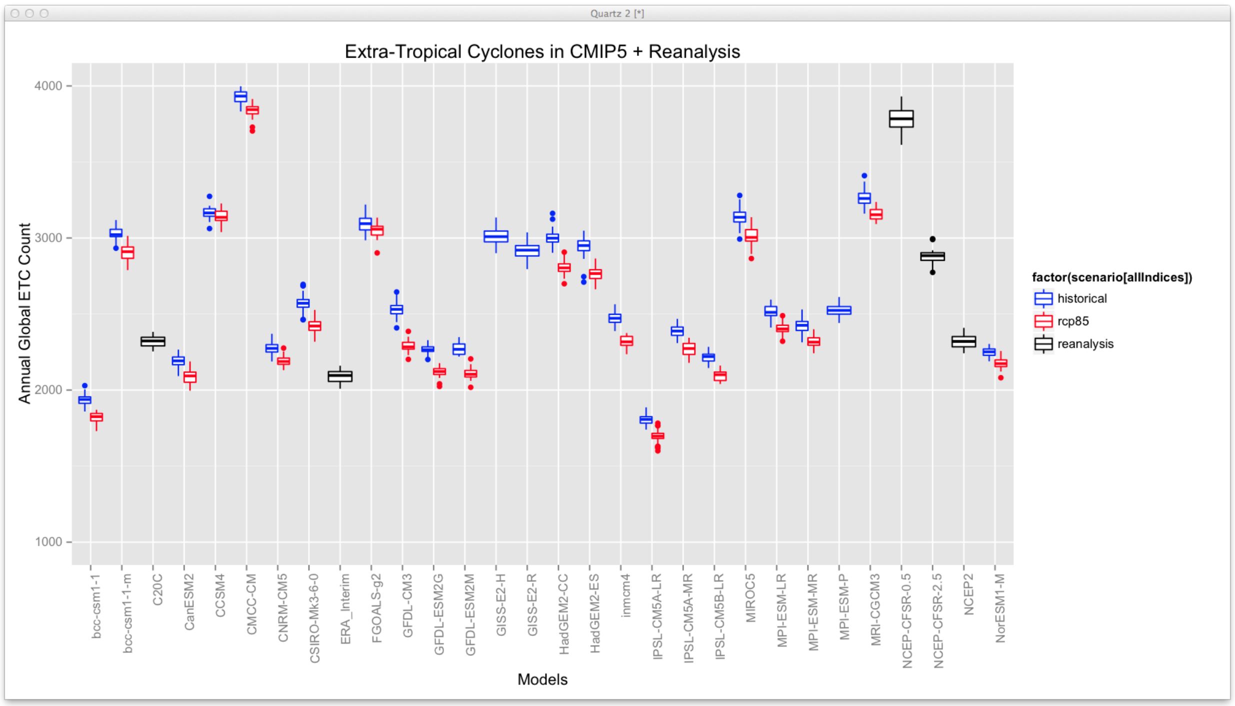


Figure 2.7: Summary of Annual Extra-Tropical Cyclone activity in all of CMIP-5. A clear decrease is observed is observed from the blue (historical) to the future rcp8.5 (red) periods

## 2.3.1 Mechanisms

Extra-Tropical cyclones are an important mechanism for transport of moisture and energy from the equator to the poles. The rcp8.5 scenario entails an uneven heating pattern, wherein the poles warm up more than the equator and the extra-tropics. Therefore the poleward surface temperature gradient actually *reduces* in the rcp8.5 scenario. We also know that there is increased stability in the subtropics due to enhanced upper-tropospheric warming. This will likely reduce baroclinic instability, potentially resulting in fewer vorticity seeds. The combination of both of these factors thereby results in fewer ETCs forming and successfully transporting energy. Further analysis exploring this gradient hypothesis could be conducted by designing aquaplanet simulations that involve uniform and uneven SST heating patterns, followed by TECA detection. We note that several other contemporaneous studies have reported a decrease in extra-tropical storm activity in North America [15, 48], and explored the poleward gradient mechanism to explain the decrease in storm counts [71].

# Chapter 3

# Changes in Extreme Precipitation

Our innovations in scalable Deep Learning, based on the ClimateNet dataset, and novel architectures for pixel-level segmentation of patterns, provides us with the unique capability to characterize extreme precipitation *conditional* on tropical cyclones and atmospheric rivers. Deep Learning and the ClimateNet project will be discussed in Chapters 4, 5 and 6. We will now present the combined application of TECA, Deep Learning and spatial statistics for the characterization of changes in extreme precipitation under global warming.

## 3.1 Tropical Cyclones

### 3.1.1 Global Tropical Cyclone Precipitation

We calculate annual precipitation from tropical cyclones across the globe for both climate scenarios, All-Hist and the so-called UNHAPPI scenario, by extracting precipitation at every pixel within TC segmentation masks from all datapoints (50 years of data from All-Hist and UNHAPPI). We note that these are average daily rainrates for a 25-km model at 3-hourly timesteps. Figure 3.1 shows how tropical cyclone precipitation intensifies and increases in a warmer world. In line with previous studies [265], we see that the PDF of TC precipitation shifts to higher rainrates under global warming. In Table 3.1 we show the percentiles and rainrates for extreme precipitation from TCs (annual and global). We compare the actual scaling of extreme precipitation with the Clausius-Clapeyron (CC) scaling rate of 7% per K, the scaling rate of available precipitable water in highly saturated atmospheres [165, 172, 3]. Notably, these studies found that global mean precipitation increases tend to be lower than the increases in the extremes due to different controlling physical mechanisms for each. The tropical (40S-40N) mean SST increase between All-Hist and UNHAPPI, is 1.6K. If extreme tropical cyclone precipitation were to follow a CC scaling relationship, we would expect about an 11.2% increase in extreme precipitation in the warmer simulation. In the last column of Table 3.1 we see that extremes at the 95th percentile and above scale at super-CC rates. These findings are consistent with hurricane extreme event attribution studies [207, 176, 169, 255] and other idealized tropical analyses [165].

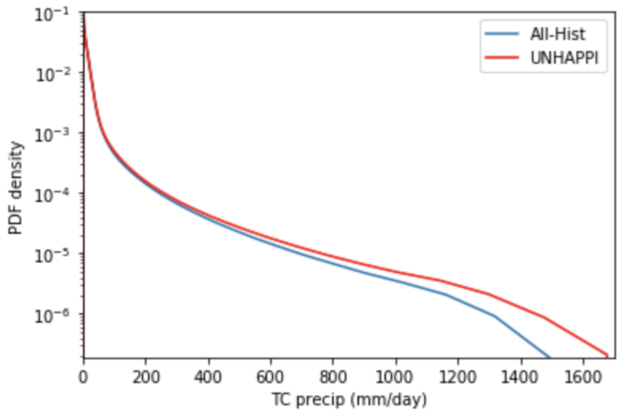


Figure 3.1: Conditional PDF for Tropical Cyclone precipitation computed using fastKDE [168]

| Scaling of TC precipitation under climate change | | | |
|---|---|---|---|
| percentile | precipitation (All-Hist) [mm/day] | precipitation (UNHAPPI) [mm/day] | % increase |
| 90 | 46 | 51 | 11.1 |
| 95 | 100 | 116 | 15.5 |
| 99 | 379 | 442 | 16.5 |
| 99.9 | 1010 | 1163 | 15.1 |
| 99.99 | 1476 | 1683 | 14.0 |

Table 3.1: Scaling relationships for global Tropical Cyclone precipitation at various percentiles for extremes. The tropical (40S-40N) mean SST increases by 1.6K from 297.4K (All-Hist) to 299.0K (UNHAPPI). CC scaling for this temperature increase would be 11.2%. Note that extreme precipitation at the 95th percentile and higher exceeds CC scaling with increases about 15%.

### 3.1.2 Tropical Cyclone Precipitation in Gulf of Mexico

Now we focus on the Gulf of Mexico and examine how TC precipitation changes in this region due to global warming. As the changes in SST are more uniform in the Gulf than they are in the entire global tropical cyclone domain, analysis of the effect of these changes on extreme precipitation is simplified. Once again, we calculate the PDFs of precipitation

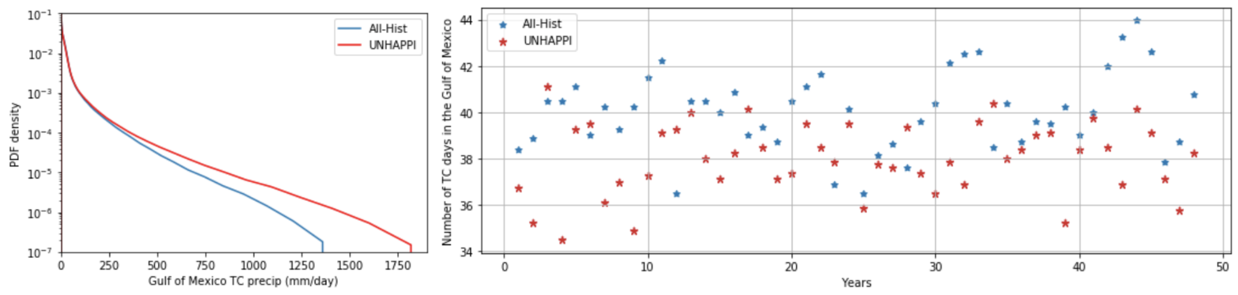


Figure 3.2: (a) Conditional PDF for Tropical Cyclone Precipitation in the Gulf of Mexico using fastKDE [168] with 50 years of data from All-Hist and 50 years of data from UNHAPPI. (b) Number of Tropical Cyclone days annually in the Gulf of Mexico, shown for 50 years of All-Hist and UNHAPPI.

| Scaling of TC precipitation in the Gulf of Mexico under climate change | |
|---|---|
| percentile | % increase (UNHAPPI vs. All-Hist) |
| 90 | 19.7 |
| 95 | 21.7 |
| 99 | 31.4 |
| 99.9 | 35.7 |
| 99.99 | 37.4 |

Table 3.2: Scaling relationships for extreme TC precipitation at various percentiles in the Gulf of Mexico. The increase in mean SSTs in the Gulf of Mexico between All-Hist (299.5K) and UNHAPPI (301.3K) is 1.8 K, which results in a CC scaling of 12.6%. We see that extreme precipitation here scales well above CC scaling, corroborated by other studies [207, 169, 255].

and changes in percentiles of extreme precipitation. The percentage increase in extreme precipitation corresponding to different percentiles are shown in Table 3.2. The increase in the Gulf of Mexico's temperature between All-Hist (299.5K) and UNHAPPI (301.3K) is 1.8 K, which corresponds to a CC scaling of 12.6%. The last column of this table shows that extreme precipitation due to TCs in the Gulf of Mexico scales well above CC, up to almost 3 times the CC scaling for the most extreme events. Table 3.2 also suggests that extreme Atlantic hurricanes that are formed in or enter the Gulf of Mexico rain much more intensely compared to global TC trends (illustrated in Figure 3.1). Further, we examine the number of TC days (defined as a day when at least one TC is active within the specified region, here, the Gulf of Mexico) in both climate scenarios. In line with what is expected for Atlantic hurricanes under global warming [265, 110], we find that, on average, the number of TC days per year decreases from 40.1 (All-Hist) to 37.7 (UNHAPPI). Hence, total precipitation increases by 18.8% per TC day, suggesting that the fewer TCs in the warmer UNHAPPI simulations produce much more precipitation than the cooler All-Hist.

## 3.2 Atmospheric Rivers

### 3.2.1 Global Atmospheric River Precipitation

Here we present annual precipitation from atmospheric rivers across the globe for both climate scenarios, All-Hist and UNHAPPI. Figure 3.3 shows how AR precipitation intensifies and increases in a warmer world. In line with previous studies [258, 46, 53, 51], we see that the PDF of AR precipitation shifts to higher rainrates.

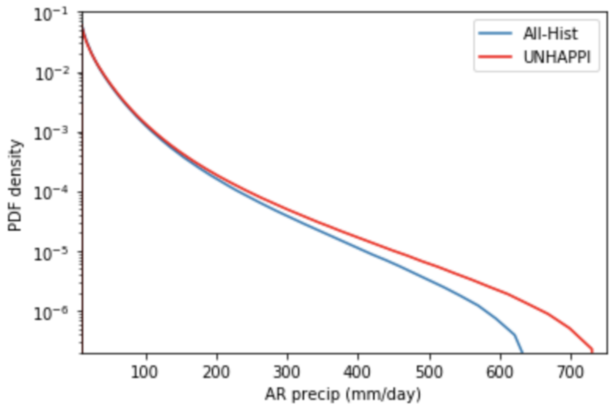


Figure 3.3: (a) Conditional PDF for Atmospheric River precipitation computed using fastKDE [168] with 25 years of data from All-Hist and 25 years of data from UNHAPPI.

In Table 3.3 we show the percentiles and rainrates for extreme precipitation from ARs (annual and global). The mean SST for AR zones (mid-latitudes, *i.e.* 30S-60S, 30N-60N) increases from 284.9K (All-Hist) to 286.6K (UNHAPPI). Hence, for this 1.7K increase in the reference temperature for ARs, CC scaling implies a 11.9% in precipitation. The actual percentage increases are shown in the last column, and we see that AR precipitation increases scale less than CC for UNHAPPI vs. All-Hist below the 99th percentile, but more than CC for the most extreme events (at and above the 99th percentile). [51] found projected increases precipitation from ARs are primarily due to thermodynamic effects controlled by CC while dynamical effects work counter to this increase for North America. Furthermore, as can be seen in the monotonic increase in scaling percentages across percentiles, precipitation in stronger ARs intensifies more than weaker ARs in a warmer world (compared to All-Hist). For comparison, [258] saw mean winter precipitation increase by 11%-18% for the west coast of North America under RCP8.5 while for extreme IVT days, which are closely associated

| Scaling of AR precipitation under climate change | | | |
|---|---|---|---|
| percentile | precipitation (All-Hist) | precipitation (UNHAPPI) | percentage increase |
| 90 | 42 | 44 | 3.6 |
| 95 | 67 | 70 | 4.3 |
| 99 | 148 | 159 | 7.6 |
| 99.9 | 336 | 378 | 12.7 |
| 99.99 | 603 | 688 | 14.1 |

Table 3.3: Scaling relationships for Atmospheric River precipitation at various percentiles for extremes. Note that extreme ARs have more extreme precip in a warmer world. The mean SST for regions where ARs are most dominant (mid-latitudes, *i.e.* 30S-60S, 30N-60N) increases by 1.7K from 284.9K (All-Hist) to 286.6K (UNHAPPI). Hence CC scaling implies a 11.9% in precipitation.

with ARs in this region, precipitation increases by 15%-39%. The findings here are consistent with [258] although our reported percentages are lower, potentially due to [258] examining winter time precipitation in California, which shows robust projected increases [237], whereas we examine across the entire year globally.

We now address two questions that highlight the power of pixel-wise segmentation in making localized, precise statements about tropical cyclones and atmospheric rivers in the USA.

### 3.2.2 Atmospheric River Precipitation in California

Next we focus on California and examine how AR precipitation changes in this region due to global warming. We choose California as ARs play a critical role in California; they can deliver 50% of the annual precipitation but also be a threat to public safety and infrastructure through extreme events [40].

Once again, we calculate the PDFs of precipitation and changes in percentiles of extreme precipitation. The percentage increase in extreme precipitation corresponding to different percentiles are shown in Table 3.4. The increase in SST off the coast of California between All-Hist (288.4K) and UNHAPPI (289.7K) is 1.3 K, corresponding to CC scaling of 9.1%. Note that for all percentiles presented in Table 3.4, we observe super-CC scaling of extreme precipitation from California ARs. These findings are similar to those of [51] and [193].

We also examine the number of AR days (defined as a day when at least one AR is active within the specified region, here, California) in both climate scenarios. We find that, on average, the number of AR days per year increases from 36.1 (All-Hist) to 37.9 (UNHAPPI) mainly due to increases in coastal SST. However, total precipitation increases by 36.9% per AR day, suggesting that west coast ARs tend to produce much more precipitation in UNHAPPI. These findings are consistent with a global analysis of ARs under climate change

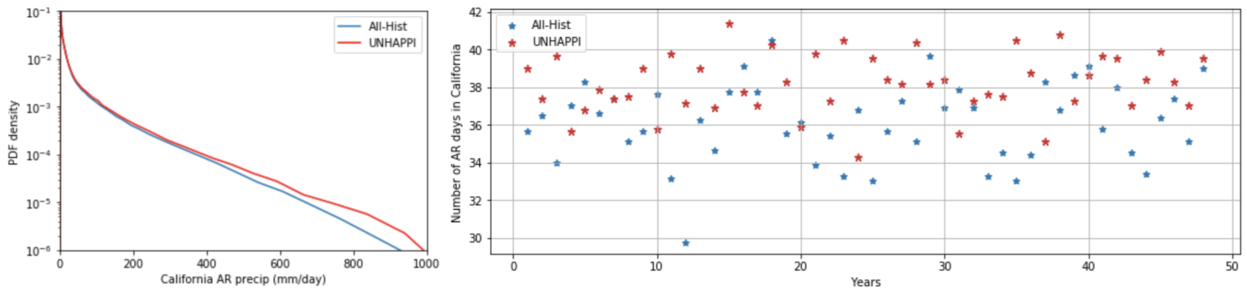


Figure 3.4: (a) Conditional PDF of Atmospheric River Precipitation in California using fastKDE [168]; (b) Number of Atmospheric River days annually in California, shown for 50 years of All-Hist and UNHAPPI.

| Scaling of AR precipitation in California under climate change | |
|---|---|
| percentile | percentage increase (UNHAPPI vs. All-Hist) |
| 90 | 12.8 |
| 95 | 11.4 |
| 99 | 11.7 |
| 99.9 | 13.2 |
| 99.99 | 10.3 |

Table 3.4: Scaling relationships for extreme AR precipitation at various percentiles in California. The increase in SST off the coast of California between All-Hist and UNHAPPI is 1.3 K, which corresponds to a CC scaling of 9.1%. Note that extreme precipitation scales at super-CC.

by [46] and regional analysis by [237]. [237] found projected increases in California's extreme precipitation event frequency. Furthermore, they found these increases to occur during the core winter months and decrease outside of these months. [46] found fewer individual AR events under climate change but an increase in AR conditions globally. Both [46] and [139] report AR conditions to increase by 50% and AR IVT strength to increase by 25%. This growth in AR conditions is linked to the increase in both size and IVT intensity of individual AR events. The change in AR days found here falls below the 50% reported by [46] and [139] as their AR condition frequency calculations were done at a grid level which is relatively more sensitive to the increased length and width of ARs compared to our metric. Regardless of AR strength or size, if it makes contact with California, it will register as an AR day.

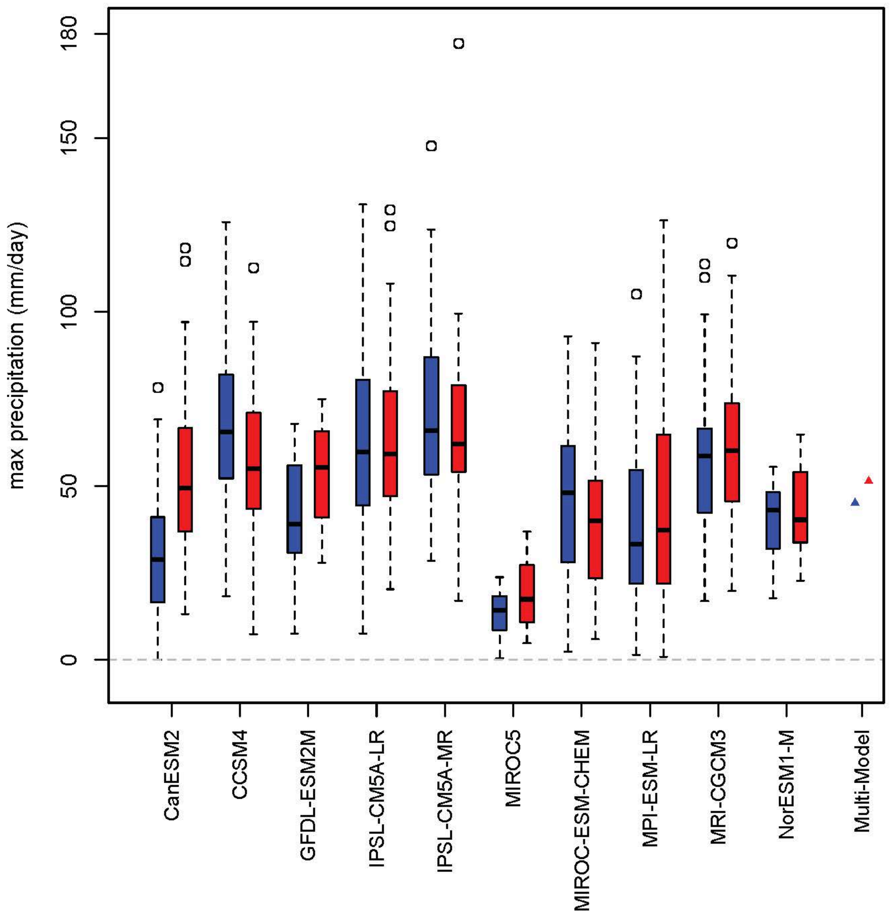


Figure 3.5: Boxplot summaries of annual maximum AR precipitation (unit: mm/day) in California from each CMIP5 model, during the 25 years spanning 1981-2005 (blue boxplot) and 2076-2100 under RCP8.5 (red boxplot). Blue and red triangles represent the ensemble mean of AR precipitation extremes for the historical and RCP8.5 runs, respectively.

### 3.2.3 Spatial Properties of Extreme AR Precipitation

We turn now to the characterization of extreme precipitation during ARs making landfall in California. Figure 3.5 shows box plots of the annual maximum AR precipitation (in mm/day) for each CMIP5 historical and future RCP8.5 runs. The AR extreme precipitation increases as temperature increases for most of the models. The ensemble mean of the annual maximum AR precipitation changes from 45.1 mm/day (blue triangle) to 51.3 mm/day (red triangle), showing a 14% increase on average. However, the individual models exhibit both increases and decreases. 5 models (CCSM4, IPSL-CM5A-LR, IPSL-CM5A-MR, MIROC-ESM-CHEM, and NorESM1-M) show decreases while the other 5 models (CanESM2, GFDL-

ESM2M, MIROC5, MPI-ESM-LR, and MRI-CGCM3) show increases in median values of this AR-related precipitation extreme.

Now we investigate changes in the spatial properties of extreme precipitation associated with ARs in a warmer climate. To find spatial variability between AR extreme precipitations, we summarize the ensemble means of maximum precipitation amounts within the AR events at each grid point. Nine grid points on the common grid are selected among all 34 grid points in California (Figure 3.6 (a))—to represent the spatial pattern of the extreme precipitation throughout southern California (stations 2, 9 and 11), central California near Sierra mountain (stations 16 and 18) and northern California (stations 23, 25, 29 and 31). In Figure 3.6 (b), we show boxplots of overall max precipitation (i.e., annual maximum precipitation) versus max precipitation associated with AR events during 25-year time periods at nine grid points in California.

Increases in maximum precipitation during AR events are consistent with a general pattern toward a warmer climate in the region, but the amount of increases vary spatially—showing relatively larger changes in the extreme rainfall amounts for northern California compared to those for southern California. Figure 3.6 (b) also illustrates the differences between extreme precipitation amounts within AR and non-AR events by grid point. We find that the median values for maximum AR precipitation are always lower than those for annual maximum precipitation in both the historical and future RCP8.5 runs. This is an indication that the AR storms identified by the TECA algorithm do not produce the most extreme precipitation events *in the models*. It is likely that extra-tropical cyclones, with cyclogenesis near the Aleutian Islands are responsible for the highest annual daily precipitation totals despite the fact that they are generally colder storms than are ARs of tropical origins. This result may not hold for the averages of all storms of these two classes because of this temperature difference. However, for extreme precipitation events, dynamical considerations can be as important as the thermodynamical. A more detailed analysis of the differences between storm types is outside the scope of this work [257, 64].

From fitting Brown-Resnick max-stable process to maximum AR precipitation, the range ($\theta_1$) and smoothing ($\theta_2$) parameters in the power law variogram are estimated from each CMIP5 model for modeling of the process; the ensemble means of those estimates, standard deviations, and 95% confidence intervals for the parameters are shown in Table 3.5. The range parameter estimates decrease in the RCP8.5 scenario relative to the present-day simulation, representing less spatially correlated AR precipitation extremes in the future. In other words, the distances, over which AR extreme precipitation amounts are correlated, are decreased, suggesting that correlation between intense AR precipitation patterns are reduced in size. However, the changes in range parameter estimation are not statistically significant under a two-sided permutation test at a 5% significance level ($p$-value=0.051).

| historical period | | | |
|---|---|---|---|
| | mean | standard deviation | 95% confidence interval |
| range, $\theta_1$ | 4.56 | (1.39) | (3.94, 5.18) |
| smoothing, $\theta_2$ | 1.47 | (0.23) | (1.36, 1.57) |

| RCP8.5 | | | |
|---|---|---|---|
| | mean | standard deviation | 95% confidence interval |
| range, $\theta_1$ | 3.72 | (1.19) | (3.18, 4.25) |
| smoothing, $\theta_2$ | 1.59 | (0.27) | (1.47, 1.71) |

Table 3.5: Multi-model ensemble means of range and smoothing parameter estimates, standard deviations, and 95% confidence intervals for parameters from modeling of Brown-Resnick processes. Range parameter estimates decrease under the future RCP8.5 scenario.

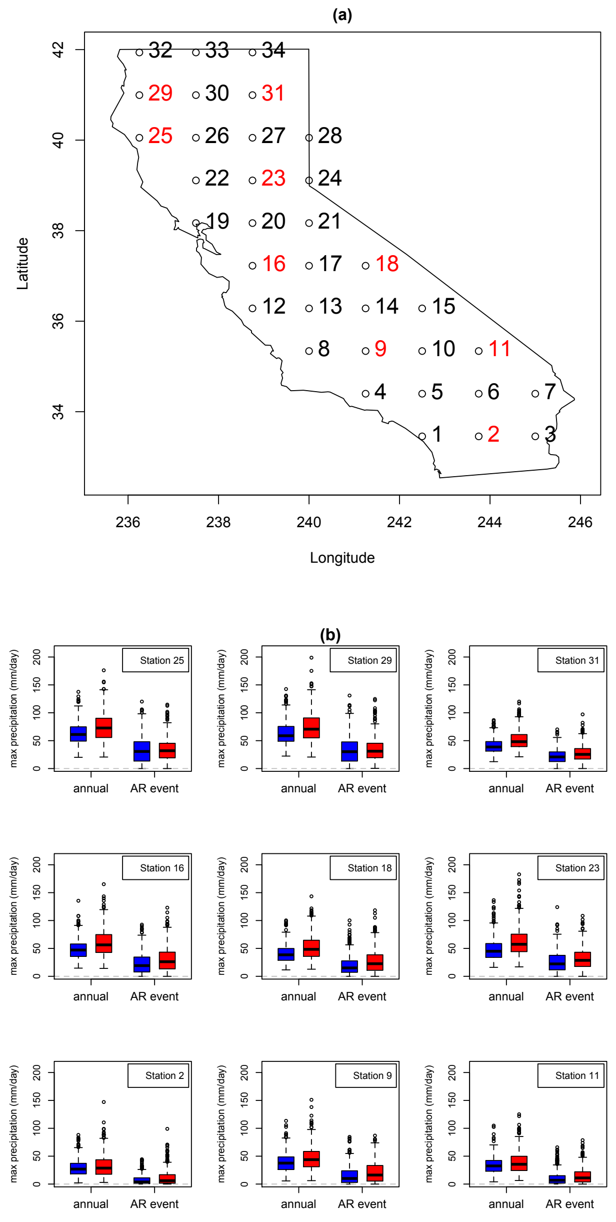


Figure 3.6: (a) Grid points of CMIP5 models in California, USA, and (b) boxplot summaries of annual max precipitation (left pair of box and whiskers at each indicated grid point) and maximum precipitation within AR events (right pair of box and whiskers at each indicated grid point) during 25-year time periods at nine grid points in California. Blue and red boxplots represent the historical and RCP8.5 runs respectively, and the station numbers are used to identify grid points in red, Figure (a).

In Figures 3.7, we illustrate changes in the spatial extent of dependence between AR precipitation extremes from multi-model ensembles. Pairwise extremal coefficients (which range from 1 to 2) are estimated between the black triangle point and the other grid points and plotted with a color gradient for the grid points 23, 16 and 9 in Figure 3.6 (a). The coefficient is 1 (green) in the case of perfect dependence and increases to 2 (tan–white) when two locations are completely independent, i.e., not spatially correlated. Extremal coefficients from the focal grid point to areas beyond the grid points are estimated using *kriging*, a method of statistical interpolation, and colors corresponding to the extremal coefficient estimates are plotted all over the California region. Differences in the ensemble average between the historical run (1981-2005) and the future run (2076-2100) are illustrated in Figure 3.7 (c). Blue and red colors represent decreases (negative values in range of $-0.17$ to 0) and increases (positive values in range of 0 to 0.02) in spatial dependence under global warming, respectively. The decreases (blue) in spatial dependence represent that AR precipitation extremes are spatially less correlated under the high future emission scenario, while the increases (red) represent more spatially correlated AR extreme precipitation than in the historical run.

The range of spatial dependence (green area) is concentrated within a smaller localized area in California in the future under the highest emission scenario than the current climate. In particular, the range of strong dependence from a focal grid point in the northern California to other points becomes narrower than the range of dependence from the focal grid points in central or southern California under RCP8.5. Though we arbitrarily selected three focal locations as representative of the three regions of California, the decreasing pattern of dependence range is true for other focal locations as well. The blue colors in the difference plots of Figures 3.7 show less spatially correlated pattern of annual maximum AR precipitation between a focal grid point to other grid points in California in a warming scenario. Small areas of grid points near focal grid points show increases (i.e., red colors in the difference plots of Figures 3.7) in spatial dependence. Therefore we postulate that future extreme precipitation during ARs will be less spatially correlated than in the current climate. However, the negative changes in multi-model ensemble means of spatial dependence are not statistically significant at the 5% significance level and the changes of dependence pattern remain uncertain.

Figure 3.8 shows lower and upper bounds of 95% confidence interval for ensemble means of pairwise extremal coefficients from both the historical and RCP8.5 runs. The confidence interval to account for the uncertainty in point estimation of extremal coefficients is obtained from 500 bootstrap samplings with all possible multi-model ensemble members and time replication. The confidence interval is narrower for areas near a focal location representing small uncertainty, whereas the confidence interval is wider in distant areas from the focal grid point. Compared to the upper bounds in Figure 3.8, trends in decreases of spatial dependence are more obvious at a focal grid point near the northern California than near the southern California.

To summarize pairwise spatial dependence visually over the entire grid, we transform the extremal coefficients to the values between 0 (complete independence) and 1 (complete dependence) by a simple calculation. We propose an arbitrary value, 1.3, as a threshold of

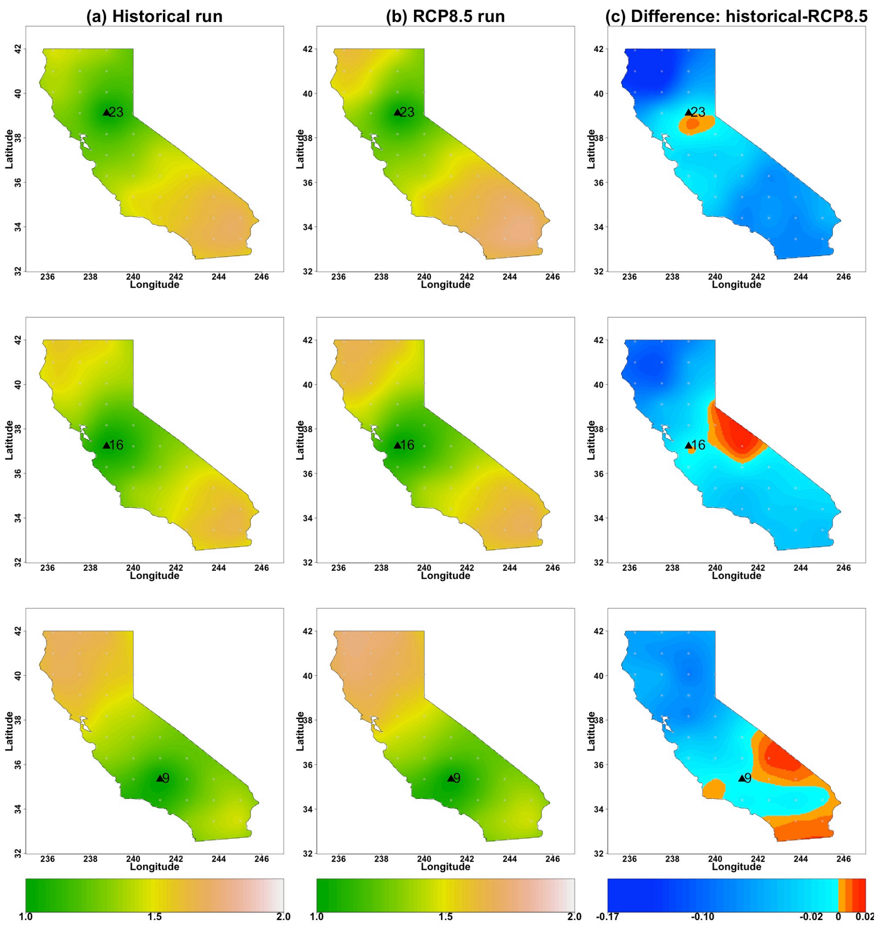


Figure 3.7: Ensemble means of pairwise extremal coefficients of annual maximum AR precipitation from a focal grid point (black triangle; grid locations 23, 16 and 9 from top to bottom) to other locations in California for CMIP5 multi-models. Changes are shown over two 25-year time periods: (a) 1981-2005, (b) 2076-2100—as well as (c) the difference.

strong dependence and count the number of values with strong dependence ($< 1.3$) at each grid point. The strong dependence ($<1.3$) shows bigger changes that are easy to capture rather than mild dependence about 1.5, and the arbitrary threshold of 1.3 is chosen as the value of strong dependence. The count takes the values between 0 and the number of grid points, not including the focal grid point in the model. Then, we normalize the count by the total number of points to be the same scale for all models, and we color the grid box based on the fraction, as shown in Figure 3.9. For example, the value 0 represents any location that does not show strong dependence (relative to the threshold) with the grid point in question. If the grid point is strongly dependent for all other 33 locations (showing extremal

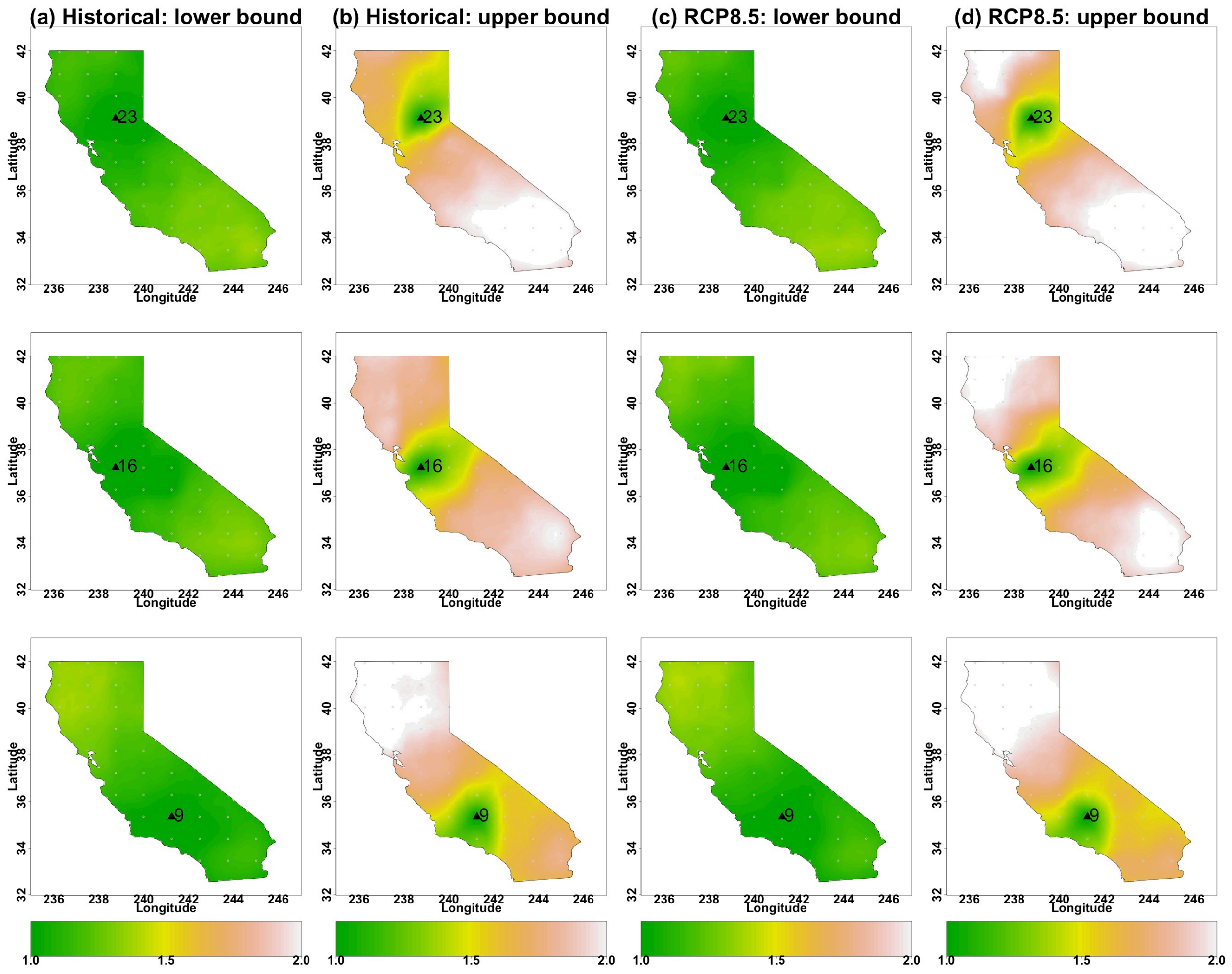


Figure 3.8: Lower ((a) for historical run and (c) RCP8.5 run) and upper bounds ((b) for historical run and (d) RCP8.5 run) of 95% confidence intervals for the ensemble means of pairwise extremal coefficients in Figure 3.7 for CMIP5 multi-model ensembles. Black triangle points represent the grid locations 23, 16 and 9 from top to bottom.

coefficients smaller than 1.3), the fraction is 1.

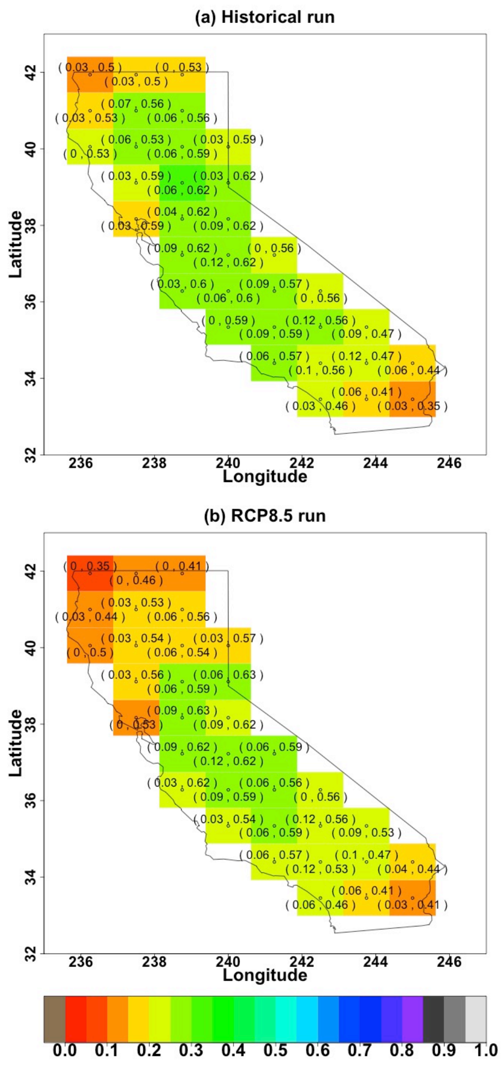


Figure 3.9: A summary of the changes in pairwise spatial dependence from (a) 1981-2005 to (b) 2076-2100 for multi-model ensemble simulations. Discrete colors at each location represent the number of locations with strong dependence (i.e., extremal coefficient< 1.3). The values in parentheses at each color box represent lower and upper bounds of the 95% confidence interval for the number of locations.

Figure 3.9 shows color changes more orangish, yellowish, and yellowish green over the region, representing more independent patterns, among maximum AR precipitation within a warmer climate. Specifically, the decreases (more orangish and yellowish color in the warmer future) in spatial dependence are obvious in northern California from multi-model ensemble means. There are also some squares near the coast and southern squares showing decreases (changes from greenish to yellowish green color) under the RCP8.5 scenario. Based on a two-sample permutation test, this analysis provides statistically significant evidence of the changes in ensemble mean of the number of locations showing strong dependence ($p$-value=0.005). The values in parentheses at each color box in Figure 3.9 represent lower and upper bounds of the 95% confidence interval for the count of locations showing strong dependence. The bounds of the interval represent the uncertainty in estimating the number of locations related to strong dependence. Here, the upper bound is more valuable to interpret than the lower bound, as a suggestion of possible limits of the values. Upper bounds in the RCP8.5 scenario are also smaller than those in historical run in northern California area.

#### 3.2.3.1 Conclusions

We have studied the influence of atmospheric rivers (ARs) on the spatial coherence of extreme precipitation under a changing climate. We have detected AR events using the TECA framework and investigated changes in properties of ARs such as total AR days, frequency, intensity of precipitation extremes associated with ARs, and spatial dependence patterns of extreme rainfall in multi-model ensemble means from CMIP5 simulations. A brief summary is provided in Table 3.6. We find there are significant increases in AR days, AR frequency, and occurrence of heavy rainfall under future RCP8.5 scenarios. We show that the spatial dependence between extreme precipitations during ARs decreases in future, warmer climates. Although future ARs produce more severe rainfall, different AR events might bring extreme rainfall intensity that is less spatially correlated under the warming scenario. However, the changes in spatial dependence are not significant, implying large uncertainty to conclude the decreases of spatial dependence between AR extreme precipitations from model outputs.

Our current analysis remains preliminary due to the limited data availability from the CMIP5 models. The higher resolution models from the HighResMIP subproject of the CMIP6 need to be considered to characterize changes in AR properties and the behavior of tail dependence across models. Furthermore, simulated extreme precipitation amounts from the relatively coarse horizontal resolutions of the CMIP5 models are substantially lower than in the real world [261]. Lavers et al. [121] discussed projected changes in ARs suggesting that these changes are thermodynamic responses to warming climate from anthropogenic radiative forcing. Further studies on the physical and dynamical processes of ARs are necessary to better understand the changes in these storms in a warmer climate AR as well as in their impacts on California.

| | historical (1981-2005) | RCP8.5 (2076-2100) | Changes (%) | 95% C.I. |
|---|---|---|---|---|
| **AR days** (days/year) | 14.5 | 19.3 | $+33$ | $(2.57, 7.07)$ |
| **AR frequency** (events/year) | 10.6 | 14.7 | $+39$ | $(2.54, 5.68)$ |
| **AR precipitation extremes** (mm/day) | 45.1 | 51.3 | $+14$ | $(0.94, 11.73)$ |
| **range of dependence** ($\theta_1$) | 4.6 | 3.7 | $-20$ | $(-1.73, 0.04)$ |

Table 3.6: Summary: Change in atmospheric river properties from multi-model ensemble outputs under late 21st century RCP8.5 forcing compared to the recent past (historical). There are increases in AR days, AR frequency, and heavy rainfall associated with ARs, while spatial dependence between the annual maximum AR precipitation decreases in the future under a warming scenario. 95% confidence interval (C.I.) is calculated for the difference in means (RCP8.5 run$-$historical run) of each AR property.

# Chapter 4

# Methods for Analyzing Extremes

## 4.1 Pattern Recognition with TECA

### 4.1.1 Collaborators

The research in this section was originally published in [188] and [186]. The collaborators were **Oliver Rübel**, **Surendra Byna**, **Kesheng Wu**, **Fuyu Li**, **Eli Dart**, **Venkat Vishwanath**, **Michael Wehner**, and **Wes Bethel**.

### 4.1.2 Introduction

Motivated by the science requirement to identify extreme weather patterns (tropical cyclones, extra-tropical cyclones, atmospheric rivers, etc) in climate datasets, we have developed TECA (Toolkit for Extreme Climate Analysis). The TECA framework uses the MapReduce computational pattern (illustrated in Figure 4.1) to enable highly scalable pattern recognition on massive climate datasets. While the climate science community has utilized different features/heuristics for extracting different extreme weather patterns, we note that the majority of the runtime is dominated by the feature detection step, which can exploit temporal and spatial parallelism to good effect. This corresponds to the 'Map' step in the MapReduce computational pattern. The specifics of implementing these flavors of parallelism are independent of the features in consideration; therefore, we designed TECA to do the heavy lifting in terms of loading data and providing spatial/temporal parallelism, allowing developers to concentrate on implementing the particulars for each feature/event type. The 'Reduce' step corresponds to collecting candidate locations (for events) and stitching them together after imposing spatio-temporal constraints. This step is typically performed on a single node, the size of the data that the 'Reduce' phase operates on is dramatically lower than the raw spatial fields corresponding to climate model output.

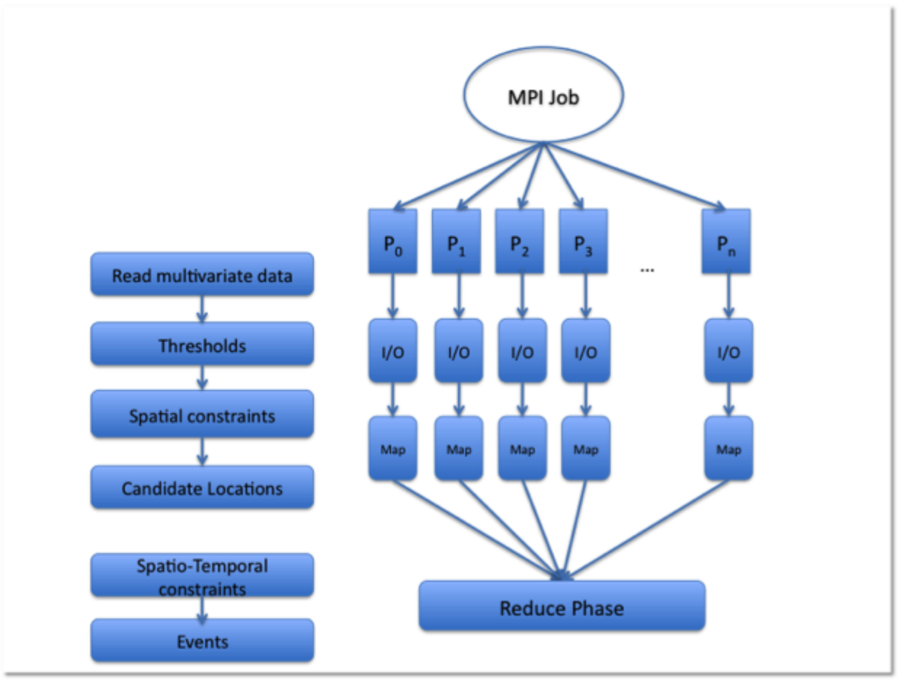


Figure 4.1: TECA utilizes the Map Reduce computational paradigm for exploiting parallel computing resources.

In designing the TECA framework, we prescribed the following goals:

- The toolkit should provide support for spatial and temporal parallelism.
- It should be easy to create an end-to-end custom analysis application.
- It should be easy to add new feature detection algorithms.
- The toolkit should provide climate-specific convenience functions (loading NetCDF files, supporting calendars, etc).

We have achieved a number of these goals in the current TECA implementation. In the following we first discuss the TECA software design (Section 4.1.3) and present three use cases that illustrate the toolkit in action (Section 4.1.4).

### 4.1.3 TECA Software Design

As outlined in the previous subsection, our design goal for TECA was to design a general framework for handling various flavors of data parallelism (spatial, temporal) and to implement common features required by every feature tracking/detection algorithm (e.g. loading NetCDF files, handling multiple calendaring systems, etc). We designed and implemented TECA in C++; the distributed features were implemented using MPI.

From the perspective of a developer who wants to implement a new feature detection and tracking algorithm in TECA, he/she needs to be able to specify a sequence of tasks and the input (data) and output (results) to/from the task. Within a task, the developer has access to a 1D/2D/3D sub-block of data and can implement the particulars of the feature detection technique. Note that the developer does not make distributed MPI calls, nor does he/she have to be bogged down with details of how the data was made available to the tasks. All he/she has to do is to iterate over their local sub-block and process the data.

Results of the detection technique are typically stored in a local table, which can be subsequently aggregated to a global table on a serial post-processing task. The developer writes custom code to analyze the contents of the table and generate the final results (statistics, tracks, etc).

We now examine the specifics of the TECA software design illustrated by Figure 4.2:

- Custom analysis tasks, such as feature detection and trajectory analysis, can be developed by inheriting from $TECA\_base\_task$. TECA already provides a set of derived task-types (e.g., $TECA\_base\_filetask$), which implement the splitting of the analysis into subtasks.

- Each task has access to local data via $TECA\_base\_data$; this includes information about local subtasks as well associated user data. $TECA\_base\_data$ is used to specify the input and output of an analysis task.

- Output from each task is typically written to $TECA\_MPI\_table$. $TECA\_MPI\_table$ implements a general distributed data table, which allows subtasks to store their results —such as cyclone locations— to the table while the table provides functionality for synchronization of the table data (i.e., gather/collect the data across MPI tasks) as well as for sorting the data records according to a user-defined subset of data dimensions (columns), e.g., yy:mm:dd.

- A custom application is designed by inheriting from $TECA\_base$ and specifying a series of tasks, and their serial or parallel execution modes.

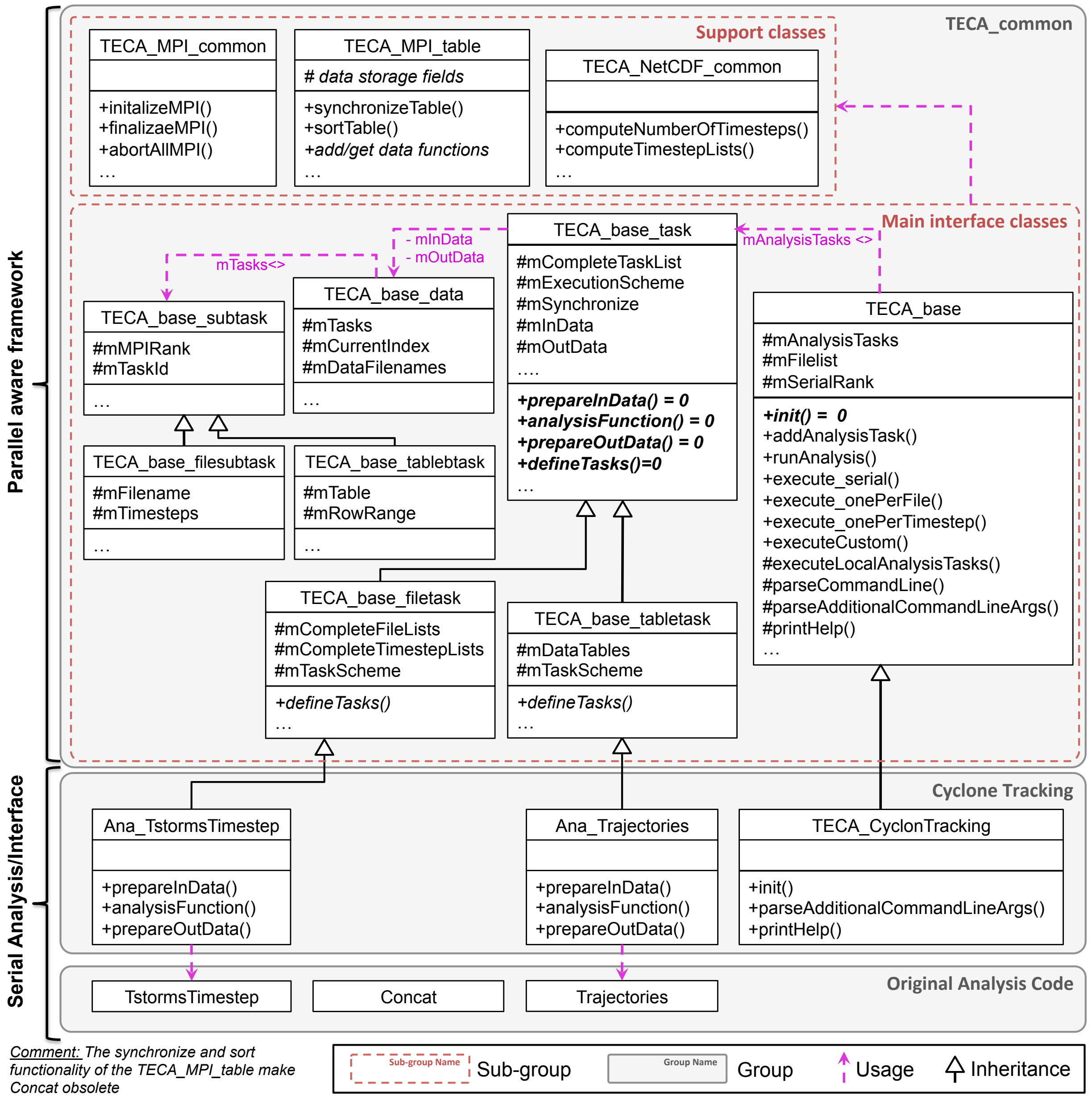


Figure 4.2: UML class diagram for the TECA software. A parallel implementation of the tropical cyclone detection is illustrated.

In the current implementation, TECA enables task classes to natively support parallelization over time (file+timestep) and data records (rows) of a data table. While it is preferable to have a 1-1 mapping between processing cores and timesteps, TECA enables users to process an arbitrary number of timesteps on each processing core. In the future, we plan to enable task classes to support spatial and spatio-temporal parallelism in addition.

Currently, interfaces for loading NetCDF metadata is provided, but individual tasks make I/O calls on their own; we will address this in the future. Support for multiple calendaring systems will also be supported in the next TECA release.

### 4.1.4 Application Case Studies

We now apply the TECA framework for three large-scale climate analysis problems, namely those of automatically detecting and tracking tropical cyclones, extra-tropical cyclones, and atmospheric rivers in climate model output. The goal of the analysis is to characterize frequency and distribution of these important extreme events. The results of such analysis are useful for validating the accuracy of climate models in reproducing these events and in investigating the changes in such events under different future scenarios.

All three case studies process high frequency output data from the Community Atmospheric Model (CAM5.1) at 0.25 degree resolution or approximately 28km at the equator. Data is stored at a temporal frequency of 3-hours. Eight time steps, corresponding to a single day, are stored in individual NetCDF files. The typical size of a single NetCDF file is 430MB; a year corresponds to roughly 156GB. Note that each analysis task needs a different subset of variables in the data files. Each test run also uses a different number of years for different analysis problems. All analysis is performed on hopper, a Cray XE6 system at NERSC. Each node of the hopper system consists of 24 AMD cores with 32GB of memory. The system uses a Seastar network as the communication interconnect, and a lustre filesystem for parallel I/O. We use a simple file-per-process I/O strategy for the reading stage.

### 4.1.5 TECA for Detecting Tropical Cyclones

#### 4.1.5.1 Algorithm

We begin our investigation by considering the TSTORMS code, originally developed at the Geophysical Fluid Dynamics Library [109] for detecting tropical storms. The TSTORMS code is representative of many such climate analysis programs: it processes climate model output one time step at a time, reads a handful of variables from the modeling output, and then performs a series of computations to produce a relatively small set of candidate points. The detection step requires the majority of the computation time and is characterized by the following conditions outlined in [109]:

1. **Vortex** is defined as the local relative vorticity maximum at the 850 hPa pressure isosurface. To be considered as a tropical storm, the maximum vorticity must exceed $1.6 \times 10^{-4}\text{s}^{-1}$, where the vorticity is the curl of wind velocity, and s denotes time in seconds.

2. **Low pressure core** is defined as a local minimum of the sea surface pressure. From the low pressure core outward, the surface pressure must increase by at least 4 hPa within a radius of 5 degrees. The local pressure minimum must be within a distance of

2 degrees latitude and longitude from the vorticity maximum. This pressure minimum is generally regarded as the storm center.

3. **Warm-core** is a local maximum of average temperature between 300 and 500 hPa. From the center of the warm-core outward, the temperature must decrease by at least 0.8 degrees Celsius in all directions within a distance of 5 degrees. The distance of the warm-core center from the storm center may not exceed 2 degrees.

All grid points that satisfy these multi-variate conditions form candidates for the next processing step. A stitching phase is now invoked, which essentially places spatio-temporal constraints involving the speed of tropical storms, the tendency to move poleward and westward, and a specified minimum temporal duration. Such criteria are used to assign candidate points to individual tracks and resolve ties. The final output are individual storm tracks which can then be counted and categorized based on maximum wind speed. Note that the computationally expensive detection step is embarassingly parallel across timesteps. The stitching step is harder to parallelize but takes up little runtime, permitting execution in a serial mode.

The original TSTORMS is a single thread sequential program written in Fortran 77. It is fairly compute intensive: serial execution on 500GB of modeling output took many days [72]. Doubling the horizontal resolution causes the detection step to require 16 times more computations. In this work, we have reimplemented the tropical storm detection algorithm in C and integrated the tropical storm detection capability into the proposed TECA toolkit.

#### 4.1.5.2 TECA integration

The design of the implementation of the cyclone detection analysis using TECA is illustrated in Figure 4.2. The initial detection of tropical storms at a single timestep is implemented in *Ana_TstormsTimestep*. The storm detection is automatically split by TECA into concurrent subtasks, each of which processes one timestep. The computation of storm trajectories is then implemented as a serial task which operates on the *TECA_MPI_table* data table generated by the storm detection.

Only minimal changes to the original analysis code are needed in order to integrate the analysis with TECA. The base analysis consists of three separate applications —Tstorms Timestep, Concat, and Trajectories— which interact through the use of intermediate files. A preliminary implementation in TECA —which preserved the communication between tasks via intermediate files— did not require any changes to the original analysis code. The implementation described here required only few changes to the analysis code to allow the storm detection (TstormsTimestep) to output its results to a *TECA_MPI_table* and enable the trajectories analysis (Trajectories) to operate on the table. Using *TECA_MPI_table* for data exchange made the Concat step obsolete and removed the need for intermediate files for data exchange.

Integration of the tropical cyclone detection with TECA required implementation of the following main user code. For each main analysis task, we defined a corresponding task class in TECA. The task classes implement the preparation of input and output data while using the original analysis code to perform the analysis task. In addition, the task classes specify how TECA should decompose the task into subtasks and how these subtasks should be executed. The main application class $TECA\_CycloneTracking$ (which inherits from $TECA\_base$) then links the different analysis tasks together to define a single coherent application. Here we mainly need to implement the $init()$ function, which instantiates the different analysis tasks and adds them to the analysis pipeline.

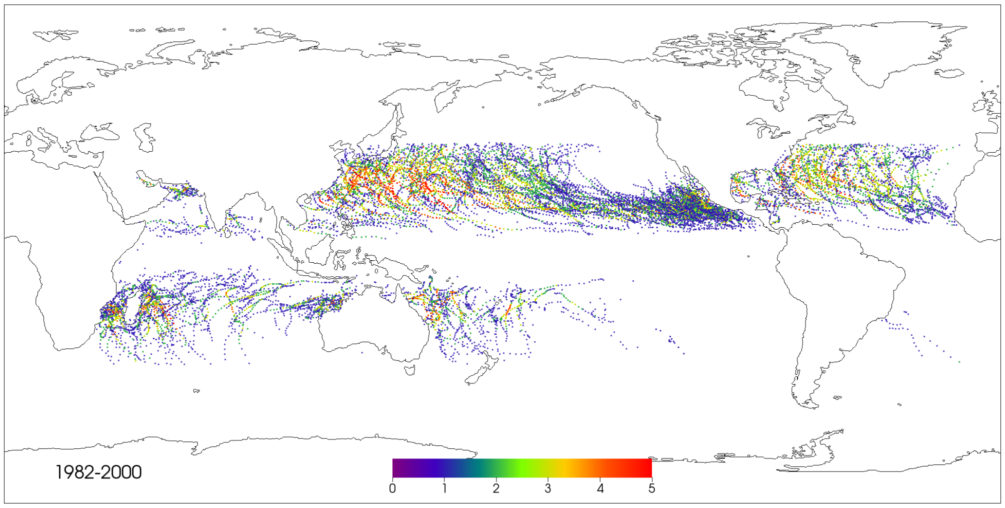


Figure 4.3: Tropical cyclone tracks for 19 years of CAM5 output. Tracks are colored by storm categories on the Sapphir-Simpson scale. Results were obtained by applying the parallel tropical cyclone analysis.

#### 4.1.5.3 Results

We processed 19 years of CAM5 output (1982-2000) with the tropical cyclone detection code. A plot of the resulting Category 1-5 trajectories is shown in Figure 4.3. We ran the detection step on 6935 cores of the hopper Cray XE6 system at NERSC. The detection step was completed in $\approx$ 2 hours. The results of the detection step was processed by the serial trajectory stitching step, which took $\approx$ 10 seconds. This result highlights the power of data parallel computing: a task that would take $\approx$ 583 days to run on completion on a single core can now be completed in a few hours.

Results from the application of the TECA TC detection capability to climate change scenarios, were discussed in Section 2.1.

### 4.1.6 TECA for Detecting Extra-Tropical Cyclones

#### 4.1.6.1 Algorithm

Our extra-tropical cyclone detection procedure is similar in spirit to cyclone detection albeit with a few differences. We follow the approach of [9, 8] for the extra-tropical cyclone detection and tracking procedure:

1. **Vortex** is defined as the local relative vorticity maximum at the 850 hPa pressure isosurface. A region of 200km × 200km is considered for determining the vorticity minima.

2. **Storm center** is defined as a closest pressure minimum within a 5 degree radius of the vortex.

A stitching step is then invoked on candidate points that satisfy these criteria. The extra-tropical storm is constrained to last for at least 2 days and travel at least 1000km over the course of its duration. A maximum distance constraint of 600km is applied for every 6-hour interval, and a preference is given to storms that travel poleward and eastward during the trajectory creation process. Similar to the tropical cyclone analysis, we found that the detection step was the most expensive.

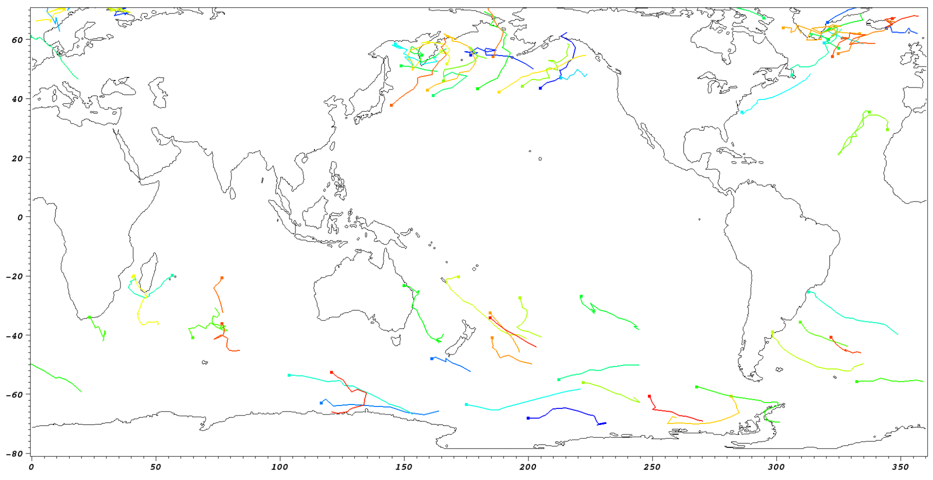


Figure 4.4: Snapshot of Extra-Tropical cyclone tracks for the month of January 2000. Tracks are colored based on individual storm systems. Results were obtained by applying the parallel extra-tropical cyclone analysis.

#### 4.1.6.2 TECA Integration

The fundamental structure of the extra-tropical cyclone detection analysis is similar to the tropical cyclone detection discussed in the previous section; both the design of the implementation as well as the effort required for integrating the analysis with TECA are comparable. Heuristics involving the detection of Atmospheric Rivers are somewhat more sophisticated in comparison, and will be described in the next section.

#### 4.1.6.3 Results

We processed 16 years of CAM5 output (1991-2005) with the extra-tropical cyclone detection code. Figure 4.4 shows sample trajectories from a single month in that duration. The processing was completed in $\approx 6$ hours on 5840 cores. The larger runtime, especially compared to the tropical cyclones, is due to the wider latitude band that much be searched for these storms, typically 20N through 90N and 20S to 90S, as compared to 30N to 30S for tropical cyclones. We note that a single core would take $\approx 1460$ days to analyze the same dataset.

The ETC detection capability in TECA was applied to a number of CMIP-5 models; the resulting changes in storm counts were discussed in Section 2.3.

### 4.1.7 TECA for Detecting Atmospheric Rivers

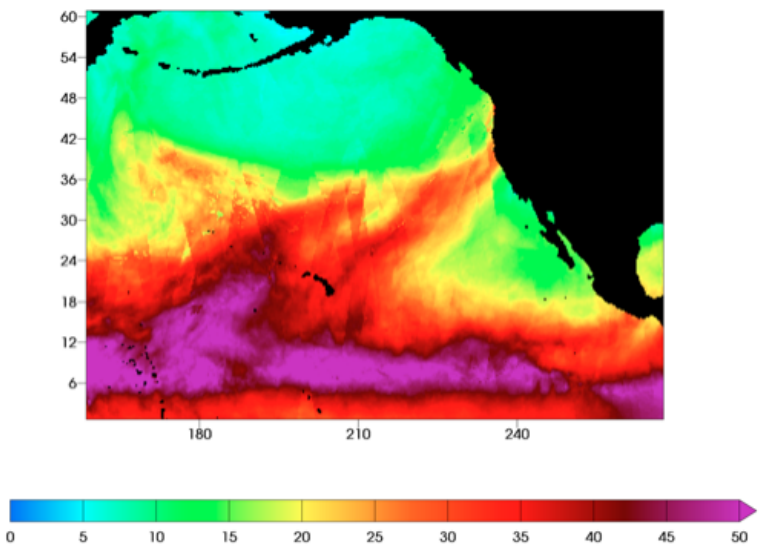


Figure 4.5: The observed three day average total precipitable water (mm) on December 14, 2010 from an analysis of SSM/I satellite data (www.rss.com). The long filamentary structure reaching west coast of US is known as an atmospheric river.

Atmospheric Rivers (ARs) are responsible for transferring moisture from the tropics to extra-tropical regions around the globe. ARs such as the one shown in Figure 4.5 passes near the Hawaiian Islands are often called the "Pineapple Express" by television weather reporters. AR events occur in oceans around the globe, including the Atlantic basin affecting the British isles. [1].

The key characteristics recognized in earlier studies of ARs is the moisture flux [282]. However, that quantity turns out to be a hard to directly observe. In 2004, Ralph et al. [198] established a much simpler set of conditions for identify atmospheric rivers in satellite observations. Their detection works with two-dimensional data over a uniform mesh on the global and is primarily based on the Integrated Water Vapor (IWV) content, which measures the total water content (measured in volume) in the volume of atmosphere above a unit of earth surface. This quantity is measured in millimeters (mm) or centimeters (cm). More specifically, they identify atmospheric rivers as atmospheric features with IWV $>$ 2cm, more than 2000 km in length and less than 1000 km in width. Based on this definition, Ralph and colleagues have identified hundreds of atmospheric river events in the data produced by Special Sensor Microwave Imager (SSM/I) satellite observations [39, 153].

In this work, we will use a different set of satellite data as well as output data from a state of the art high-resolution climate model. The observational data we use is from a satellite called Advanced Microwave Scanning Radiometer (AMSR-R). This device measures IWV allowing us to use the same conditions as proposed by Ralph et al. [198].

Objective identification of atmospheric river events is a challenging task. Identifying observed events in the historical record for case study analyses can exploit associated information such as on-shore extreme precipitation and wind direction to identify candidate events. Large scale structural information can then be gained by analyses of satellite measurements [199]. However, analyses of the statistical behavior of atmospheric rivers are also necessary to understand the more general relationship to large-scale climatic variations. The ability of climate models to simulate atmospheric river statistics is key to projecting if these phenomena change as the climate warms. Hence, an atmospheric river identification scheme that neither misidentifies nor misses candidate events is critical to the statistical analysis of climate models, and their comparison to the observed recent past.

**4.1.7.0.1 Climate Models** Using computers to simulate and predict the weather is one of the most successful applications of computer technology in the past few decades. With this technology, we are able forecast severe weather and effectively evacuate the affected area. The same technology has also been used to assess how the climate changes will impact human society in the future [170].

The Community Earth System Model (CESM) [2] is a fully-coupled, global climate model developed by the National Center for Atmospheric Research (NCAR) in Boulder, Colorado [3].

[1] http://cimss.ssec.wisc.edu/goes/blog/archives/3838.
[2] http://www.cesm.ucar.edu/.
[3] http://ncar.ucar.edu/.

It is one of the widely used global climate modeling tool used for climate research. In our study of atmospheric rivers, we will be using the output from a component of CESM called fvCAM, the finite-volume version of the Community Atmosphere Model.

Data produced by climate models and various satellites orbiting the earth is massive. For example, 15-year modeling data from fvCAM with output every 6 simulated hours, and a mesh point resolution of 0.5° latitude by 0.625° longitude produces roughly 500GB of data [262]. Ensembles of simulations can easily produce petabytes of data. The Advanced Microwave Scanning Radiometer (AMSR-E) satellite produces 150GB data for a decade of observations. There are many satellites scanning the earth for different observations. Together they produce many terabytes of data.

In this work, a key goal is to develop an algorithm for identifying atmospheric rivers from such datasets. To work with petabytes of data, our algorithm has to be efficient and able to take advantage of massively parallel computers.

**4.1.7.0.2 Feature Detection on Mesh Data** Climate Model and satellite output are typically generated (or regrided) on a regular mesh over the globe. Following the methodology used by Ralph et al., we perform our detection on 2-D data on the latitude-longitude mesh [198]. An atmospheric river is an event that can last for a few days. Our detection algorithm processes one day at a time. For each day's data, the AR appears as a connected regions in space where the integrated water vapor content is high. This type of feature in space is commonly known as *region of interest*. Identifying such regions of interest is a basic operation in many computer vision and visual analysis tasks [159, 58].

Our detection algorithm proceeds in three steps. The first step performs a thresholding operation based on IWV value; mesh points with high IWV values are marked for further processing. The second step connects the marked mesh points into regions. This step employs a connected component labeling algorithm. The connected regions are passed to the last step for verification of sizes. The first and last steps are relatively straightforward and can be found in many textbooks [58, 159]. In this section, we briefly review the algorithms used for connected component labeling [41, 271].

The IWV data processed by our feature identification procedure is stored as a 2-dimensional array. The output from the thresholding step can be treated as a binary image, where the foreground pixels are mesh points with large IWV values and the background pixels are mesh points with small IWV values. This allows us to use the connected component labeling algorithms developed from image processing. There are a variety of algorithms for this task. For example, there are a number of different parallel approaches [63, 254], some methods using specialized hardware [49, 127], and some using GPU-type accelerators [73]. In our application, the image sizes are relatively modest. For example, at 0.5° resolution, the whole global is divided into a mesh of $720 \times 360$, which has just about a quarter of a million mesh points (or pixels in the binary image). Due to this modest size, we choose to perform connected component labeling using only a single CPU core.

The sequential labeling algorithms can be divided into three categories based on how many times the binary image is accessed [236, 271]. The simplest algorithms requires multiple passes through the binary image and are known as multi-pass algorithms. The algorithm by Suzuki et al. [236] is an efficient example in this category. The second category of labeling method is known as two-pass algorithms because they require two passes through the binary image, once to gather the connectivity information among the foreground pixels and once to assign the final labels to each pixel. The third category of labeling methods requires only a single pass through the data [16]. On modern CPUs, memory accesses typically dominate the cost of the connected component labeling algorithms. In this case, the algorithm that scan through the data array the least number of times should be the fastest. This argument favors the one-pass algorithms. However, the one-pass algorithms have to perform random accesses, which are typically much slower than sequential memory accesses. For this reason, an efficient two-pass algorithm can actually outperform the best of the one-pass algorithms [271]. For this reason, we choose to use a two-pass algorithm in this work.

The two-pass algorithms need some data structures to record the equivalence information among the provisional labels assigned to the foreground pixels during the first pass. They avoid scanning the image multiple times by manipulating the label equivalence information to arrive at a final assignment for each provisional label. The most efficient data structure for keeping track of the label equivalence information is called union-find [232], and the most efficient implementation of the union-find data structure is an implicit data structure that uses a single array [271]. An efficient union-find implementation is critical to the overall effectiveness of the two-pass algorithm.

It is possible to represent the binary image differently to achieve better performance for the connected component labeling step [272]. The key idea is to make the representation of the foreground pixels more compact. However, representing the foreground pixels more compactly will make it more difficult to compute lengths and widths of the connected regions, which make the third step more expensive. In this work, we chose to keep the binary image in an two-dimensional array.

#### 4.1.7.1 Our Approach

Our algorithm processes 2-D meshes defined over the globe. These meshes are relatively small, for example, the satellite observation data is defined on a $1/4°$ mesh with just over 1M mesh points, and the climate model output uses a $1/2°$ mesh. Even with fine meshes at $1/10°$ mesh, the data associated with a single variable, i.e. integrated water vapor (IWV), can easily fit into main memory. While we need to process many timesteps in the complete dataset, this can be done in parallel.

A schematic illustration of the parallel algorithm for AR detection is shown in Figure 4.6. The algorithm can be divided into an I/O phase and and a Compute phase (shown as C in Figure 4.6). The I/O phase includes reading the input filenames and vapor data. The computation phase consists of thresholding, connected component labeling, and verification steps. Each process generates an output indicating the presence or absence of an AR.

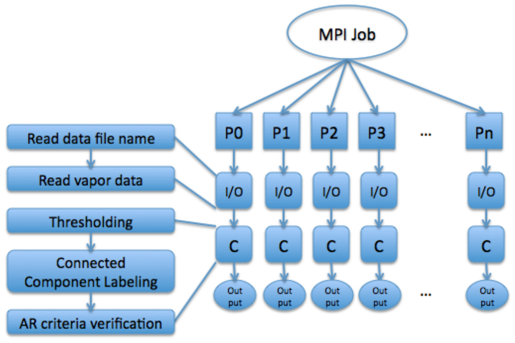


Figure 4.6: Schematics of AR detection tool implemented with MPI

Our design allows each process to run independently without any need for inter-process synchronization or communication.

**4.1.7.1.1 I/O Phase** Our current implementation requires a list of data file names to process. This list is currently stored in a single, shared file. Currently, all processes read the file; it is possible to split this file in the future to reduce metadata overhead, but for now we have decided to use this simple approach. Once each process determines what file to process, it then proceeds with reading the IWV data.

The function that performs the reading of IWV data takes a number of optional input parameters, such as granularity of climate data, type of data format (such as gunzip compressed format, netCDF, etc.), the number of timesteps present in one day's data and regions where AR should be detected. This flexibility allows us to detect AR in any region of the world at different granularity.

#### 4.1.7.1.2 Compute Phase

**Thresholding** In the compute phase, the first step is a set of thresholding operations on IWV values. Ralph et al. [198] specify IWV values greater than 20mm for detecting atmospheric rivers. We use this threshold value for all results reported in the paper. However, our detection tool can take on a pair of thresholds that define the a lower bound and a upper bound for the IWV values. This additional flexibility can be useful for with systematic biases. The output of the thresholding step is a collection of mesh points that satisfy the threshold criteria. These "foreground" pixels are then processed by the Connected Component Labeling (CCL) step.

**Connected Component Labeling** Our connected component labeling implementation is based on a two-pass algorithm [271]. The algorithm can be broken down into three steps. The first step assigns a provisional label to each mesh point visited. These provisional labels may turn out to be assigned to connected mesh points. We say that these labels are equivalent. This label equivalence information is recorded in a data structure called union-find. The second step works with the union-find data structure to determine the final label for each provision label. The third step replaces the provisional labels with their final values. This third step is a series of straightforward assignments.

The first step examines each mesh point in turn. A mesh point failing the thresholding conditions will receive a special label, say 0, to indicate that it is not of interest. A mesh point satisfying the thresholding conditions will receive a provisional label. This assignment proceeds as follows. If there is no neighbor with a provisional label already, then this mesh point receives a new label. If any of its neighbors have already received a label, any of their labels can be assigned to the current mesh point. Because the neighbors are connected to this mesh point and to each other, their labels should be the same. We say that these labels are equivalent, and choose the smallest labels as the "representative" of the groups of equivalent labels.

The union-find data structure stores the label equivalence information. This data structure supports two key operations called union and find. Given any provisional label, the find operation locates its "representative." Given any two provisional labels, the union operation is to record that they are equivalent to each other. This operation can be implemented as two find operations followed by an operation to set one "representative" pointing to the other. We choose to have the representative with larger numerical value pointing to the representative with smaller value. The union-find data structure can be interpreted as representing a forest of union-find trees, where the "representative" is the root of each tree. Pictorially, this is illustrated in Figure 4.7. By choosing to use non-negative integers as labels, it is possible to use the labels as the array index and implement the union-find data structure in a single array as illustrated in Figure 4.7.

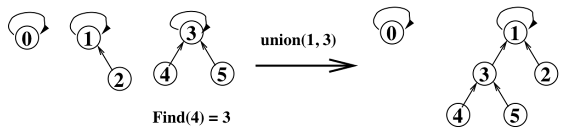


Figure 4.7: An array representation of the rooted trees.

Using an array to implicitly represent the union-find trees has the advantage that the memory for the union-find data structure is consecutive in memory. Furthermore, the find operations always traverse to the left in Figure 4.7. This predictable pattern reduces the average cost of the memory accesses, which improves the overall effectiveness of the labeling algorithm.

**Verification** After the connected component labeling step, each connected group of mesh points receives a unique label for identification. We then compute the length and width of each group, and impose the relevant constraints (i.e. $Length > 2000km$ and $Width < 1000km$ [198]) in the verification step.

Our implementation can also impose additional spatial constraints. For example, to declare an atmospheric river as a "pineapple express", we can test for the AR having passed through the islands of Hawaii and the west coast of US.

#### 4.1.7.2 Experimental Methodology

Thus far, we have described the algorithm for detecting atmospheric rivers. We now evaluate the performance of our implementation.

**4.1.7.2.1 Accuracy of our approach** We have taken a number of approaches to validate the accuracy of our detection algorithm. First, we made sure that our algorithm can accept the parameters established by Ralph et al. [198]. Theoretically, this should ensure we have the same detection algorithm as used by Ralph and colleagues.

To start off, we verified that we were able to detect recent "pineapple express" events in the recent news such the one around December 14, 2010 shown in Figure 4.5. This approach is useful but limited in that we only have established reports for a handful of such events. We considered mining weather reports for more occurrences of this pattern; however, we are not able to find a reliable mining tool to process such information.

We also used a subjective approach, wherein we had a domain science expert examine a number of "hard" cases and report if the algorithm was computing the right result. While this exercise was extremely insightful in terms of understanding the phenomena, we could not be absolutely certain that we had captured all occurrences. Needless to say, it was impossible to have the domain expert sift through all of the observational data due to time constraints.

We briefly considered the option of using a system like the Amazon Mechanical Turk[4] for obtaining human annotated data, but there is a certain degree of domain knowledge required to successfully complete this task.

Finally, we settled on comparing our results to the published AR events in the west coast US by a number of other researchers [39, 153]. These papers contain an exhaustive list of atmospheric rivers reaching the US west coast from the year 1998-2008. We treat the results reported in Dettinger et al. from June 2002 and 2008 as ground truth. We note that our results are obtained from a different satellite, Advanced Microwave Scanning Radiometer (AMSR-E) satellite [5].

**4.1.7.2.2 Hardware Platform** We conducted our experiments on the NERSC Cray XE6 supercomputing system Hopper [6]. The system has ≈6,400 compute nodes, with 24

[4]http://aws.amazon.com/mturk/.

[5]Web URL is http://www.ssmi.com/

[6]http://www.nersc.gov/nusers/systems/hopper2/

cores (total ≈150,000 cores, 2 twelve-core AMD 'MagnyCours' 2.1 GHz processors per node) and 32GB memory per node. We used all 24 cores of a node for our tests and have one MPI process on each core. Hopper uses Lustre as its file system, with a peak theoretical I/O bandwidth of 35GB/s. The parallel file system is configured with 156 Object Storage Targets (OSTs).

#### 4.1.7.2.3 Data

**Observational Data** We use a geophysical dataset derived from observations collected by the AMSR-E satellite. The overall dataset contains sea surface temperature, surface wind speed, atmospheric water vapor, cloud liquid water, and rainfall rate. The orbital data of the satellite is mapped to 0.25° mesh, i.e. each of the data observations is gridded onto a 1440 x 720 matrix. The daily data collected by AMSR-E contains gaps because the satellite can not cover the whole globe in a day. To obtain complete data for any given day, RSS provides time-averaged data using a 3-day moving window.

In our atmospheric river detection scheme, we use the vertically integrated water vapor data from files containing 3-day averages of column integrated water vapor. The files are compressed into gzipped format (.gz). We converted this compressed files into netCDF format. The size of each 3-day average file in netCDF format is 40 MB. In our tests, we used observation data for 3100 days, which amount to 124 GB. This dataset is used for verifying the accuracy of our tool in detecting atmospheric rivers in the coastal areas of California, Oregon, and Washington states. We compare the results with the manually identified list of AR events by Dettinger et al [39].

**Model Data** We use climate data generated by the finite volume version of the Community Atmospheric Model (fvCAM) in our scalability study [262]. The fvCAM uses a finite volume approximation to the atmospheric equations of motion and have been specifically optimized for parallel execution. Output data in netCDF format includes multiple variables such as pressure, humidity, temperature, total vertically integrated water vapor. For detecting atmospheric rivers, we use the data value for the integrated water vapor. The data is arranged in a 361 x 576 mesh, which represents 0.5° latitude by 0.625° longitude. There are 4 simulated time steps per one day, i.e. one per every 6 hours, and the total dataset contains 15 simulated years worth of data that amounts nearly 450 GB. The dataset is stored into 1095 files and each file consists of 5 days worth of data.

To avoid dealing with intra-day variations, our detection algorithm works with daily averages calculate from the 6-hour timesteps within the day. Since model data does not have any missing data, we did not need to compute the average for 3 days as in the observational data. In our strong scaling tests, we used data related to 10,000 days, which is ≈1 TB. In the weak scaling experiments, the data size is increased in proportion with the number of processes used. In these, each MPI process analyzes data related to one day. For example, in a 10,000 process MPI job, the application processes 10,000 days worth of data, which is in the range of 1 TB. Similar to the observational data analysis, in both weak scaling and strong scaling studies, we analyzed data related coastal areas of California, Oregon, and Washington states. The tool can be used for detecting AR in any region, by changing the longitude and latitude bounding box parameters and the AR detection criteria.

#### 4.1.7.3 Results

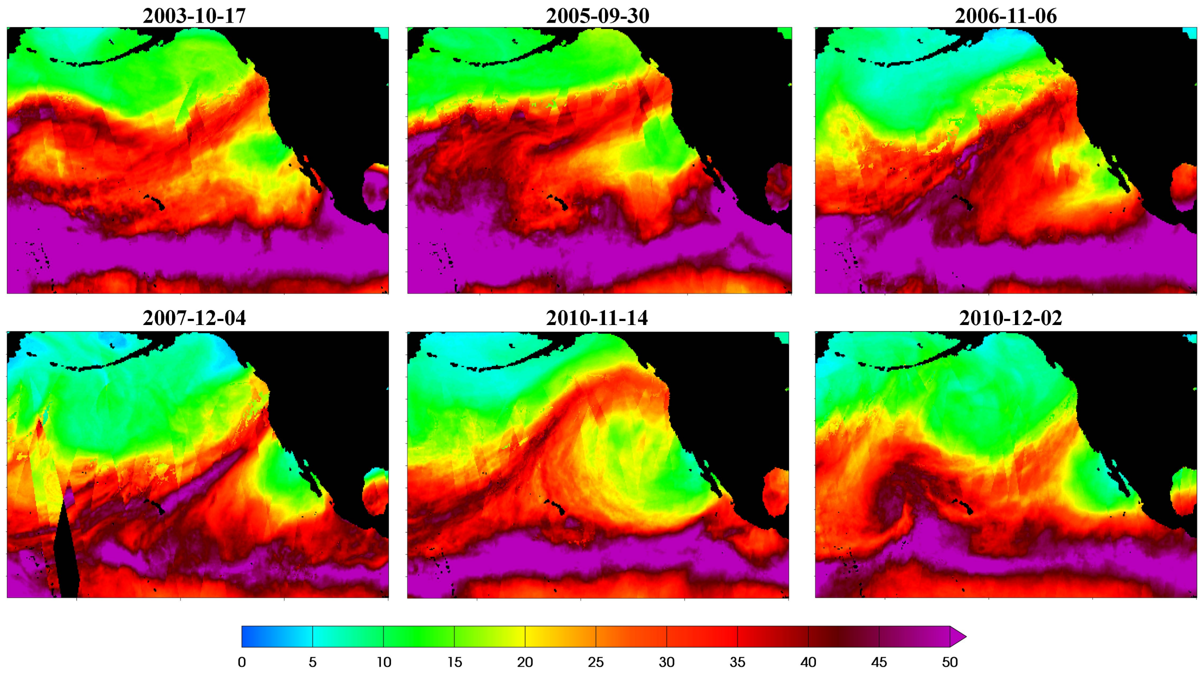


Figure 4.8: Some typical atmospheric river events detected by our new method from the observational dataset. Shown is total column integrated precipitable water in mm. Note that the structure of each event is unique. Also note that data irregularities in the satellite measurements (seen as abrupt discontinuities e.g. in the 2007-12-04 event) do not have an adverse effect on the detection procedure.

We applied our AR detection tool to the observational data, and compare the detected events with the published paper by Dettinger et al [39]. We use the same thresholds listed in their paper: water vapor (>20mm), length (>2000km) and width (<1000km), and spatial constraints of examining ARs originating in the tropics and making landfall on the western US coast. Figure 4.8 shows a sampling of detections from our program.

Our tool is able to detect 81% of the AR events reported in Dettinger, et al. Upon further examination, we discovered that Dettinger, et al. were reporting ARs even if the river did not actually make landfall (but was close to it). Our algorithm declares an AR only if the river touches the land. We thereafter removed entries from Dettinger, et al.'s list that did not make landfall. The resulting accuracy of our tool is 92%. The remaining undetected events have vapor below threshold in some parts of the narrow band, which CCL algorithm counts as different connected components. Since these disconnected labels do not fit in the source, destination, and length criteria, they are not detected as AR by the tool. We will address this issue in future refinements of our implementation.

Figure 4.9 shows statistics of AR events between 2002 and 2010. For year 2002, the data is available from June to December, and for all other years, the events are for the whole year. We counted consecutive days with an AR as one event. We separate the AR events in the winter-time from summer months. This relative distribution is quite similar to those reported in earlier studies [39, 153].

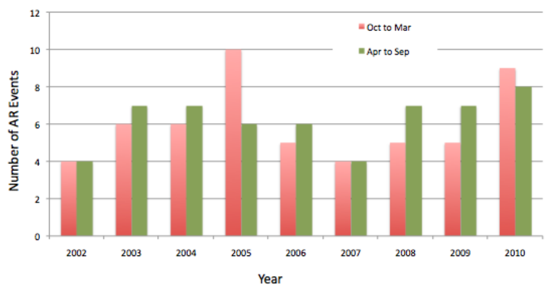


Figure 4.9: Yearly statistics of Atmospheric river events from REMSS observational data.

## 4.1.8 Related Efforts

Climate scientists typically develop specialized data analysis tools for their research problems. The tropical storm detection code [109] and the atmospheric river detection code [39, 13] mentioned earlier belong to this category. The work presented in this paper attempts to develop a parallel framework for different analysis codes to work together on top of a common substrate that can handle flavors of spatial and temporal parallelism.

There have been a number of efforts to develop more generic climate analysis tools. We review a number of existing software packages and contrast them with our present work.

CDAT (Climate Data Analysis Tool) is a powerful suite of analysis and visualization capabilities targeted for the climate community. Its stated goal is to use an open-source, object-oriented, easy-to-learn scripting language (Python) to link together separate software subsystems and packages to form an integrated environment for data analysis. The tool is well designed and has a large number of users. However, we believe that their approach does not emphasize the need for parallel data analysis. We also note that the next generation UV-CDAT [245] infrastructure is trying to address some of these limitations.

GrADS (Grid Analysis and Display System) is an interactive desktop tool for easy access, manipulation, and visualization of earth science data; the openGrADS project intends to develop advanced interfaces and extensions for GrADS. They emphasize ease of use on desktop computers and do not directly address the pressing issue of handling massive amounts of climate data.

Statistical Tool for Extreme Climate Analysis (STECA) provides a graphical user interface for users to compute statistics of daily atmospheric climate analysis, such as heat index, relative humidity, wind chill, etc. The GUI allows users to specify data files and select statistical functions. The extReams toolkit generates daily and seasonal statistics of weather. In contrast to both these tools, TECA detects extreme climate events in large climate datasets by exploiting parallelism on supercomputing platforms.

The NCAR Command Language [152] and NetCDF Operators [155] are powerful, general utilities for performing a range of operations (examining, subsetting, averaging, plotting) on climate data. Efforts such as ParCAL [173] and Pagoda [171] are developing parallel analogues of these utilities, enabling parallel processing on large datasets. Our effort is fairly specialized in this regard; we are developing a framework for parallel feature tracking and analysis.

Kendall et al. [101] present the DStep system which is similar in spirit to our work. Their system allows for simplified traversal of spatio-temporal domains across a large scale distributed memory system. They use a MapReduce-like framework to process data and merge the results. They also use sophisticated data management and parallel I/O techniques, and demonstrate teleconnection analysis on an ensemble dataset.

Finally, we note that a few visualization and analysis tools have been demonstrated to scale on large supercomputing platforms and process massive datasets. VisIt [247] and Paraview [174] belong to this category. However, these tools are designed to be general purpose and are not specifically targeted for the climate community. It is possible to implement the features in TECA within the VTK pipelines of VisIt and Paraview, and expose some climate-specific features to users. We will explore this route in the future as a vehicle for production deployment to a broad user community.

### 4.1.9 Discussion and Future Work

Using TECA greatly simplifies the parallel implementation of the tropical storm, extratropical cyclone, and atmospheric river detection analyses described in the previous section. TECA handles many of the common tasks involved with parallelizing the analysis, in particular: i) splitting of analysis tasks into independent subtasks, ii) execution of the subtasks in parallel, iii) collecting data from all MPI ranks and sorting of the collected data by using TECA's parallel table data structure for storage and exchange of analysis results, and iv) integration of different analysis tasks (e.g,. detection of storms and computation of storm trajectories) into a single analysis application. The object-oriented design of TECA provides developers with flexibility and ease of integration of new features, such as new schemes for decomposition of analysis into subtasks, without having to modify TECA itself.

The current implementation of TECA achieves a number of the goals we had proposed, such as ease of use and native support for temporal parallelism, but it also has various limitations. While a developer may easily add new types of tasks and task-decompositions, TECA currently does not provide native support for spatial parallelism, i.e., TECA currently does not implement subtasks based on spatial sub-regions of the globe. This will likely be a limitation when data from kilometer-scale atmospheric models becomes available. Simple forms of temporal parallelism, namely assigning one timestep per core, will introduce load balancing problems, wherein only a small subset of the cores will undertake a detailed search for sparse extreme events. This problem can be alleviated by assigning interleaved timesteps (or spatial subsets) across cores or a more elaborate workload-driven dynamic load balancing scheme. Finally, TECA has little I/O support at the moment. TECA currently only provides a straightforward interface to climate data, while I/O operations are largely performed by the analysis algorithms. We plan to address these limitations in future work.

Our current performance numbers reflect tests at a single problem configuration: $\approx O(6000)$-way concurrency and primarily utilizing temporal parallelism. We would like to perform more extensive scalability tests by utilizing spatial, temporal and spatio-temporal parallelism across a variable number of cores. Mixing different parallel modalities will enable us to run at much larger concurrencies and will likely expose scalability bottlenecks.

### 4.1.10 Conclusion

We have presented a design for TECA that leverages different flavors of data parallelism currently available in large climate datasets. We have demonstrated the power of the toolkit by application to three important climate science problems, namely that of detecting tropical cyclones, extra-tropical cyclones, and atmospheric rivers. We have tested these implementations on large TB-sized climate datasets and demonstrated good runtime performance on thousands of cores. The TECA toolkit provides for an extensible framework that can be leveraged for implementing different feature detection and tracking algorithms for tackling large-scale climate science problems.

Results from the application of the TECA AR detection capability to climate change scenarios was presented in Section 2.2.

## 4.2 Spatial Statistics

### 4.2.1 Collaborators

The research in this section was originally published in [94]. The collaborators were **Soyoung Jeon**, **Surendra Byna**, **Junmin Gu**, **William D. Collins**, and **Michael F. Wehner**.

### 4.2.2 Introduction

Atmospheric rivers (ARs) are narrow bands of elevated tropospheric water vapor that are thousands of kilometers long and hundreds of kilometers wide. ARs have generated recent interest because they have proven capable of producing extreme precipitation and flood damage over the western coastal United States. Ralph et al. [195] define ARs in California as containing high amounts of water vapor and strong winds that force this vapor to higher altitudes in coastal or mountainous terrain resulting in heavy precipitation. Atmospheric rivers often result in extreme winds and precipitation. For instance, Ralph et al. [197] document a specific AR event that produced more than 10 inches of rainfall in two days and caused extensive flooding near the Russian River in northern California. One particularly damaging class of ARs, popularly called the "Pineapple Express", has been implicated in several recent major storms along the west coast of the U.S. [38, 154, 37].

In order to collect statistics on AR events, we implemented a pattern detection scheme in TECA (Toolkit for Extreme Climate Analysis) [187, 13]. Using high performance computing platforms, TECA enables efficient extraction of climate patterns in large data archives such as that generated by the Coupled Model Intercomparison Project Phase 5 (CMIP5). In this section, we implemented the AR detection procedure described by existing literature [196, 154, 270] and applied it to historical and future simulations in CMIP5. The detection results presented in [270] are fairly convincing, and we did not feel the necessity to innovate further on the detection procedure. Instead, we focused on the large-scale application of the existing technique in TECA and on the characterization of the extreme precipitation resulting from the simulated ARs.

There are relatively few studies concerning the spatial variability of precipitation caused by the landfall of ARs and the impact of ARs on extreme precipitation under a changing climate. Leung and Qian [125] investigated the influence of ARs with respect to the spatial distribution of precipitation for two events, Presidents' Day (U.S.) in 1986 and New Year's Day in 1997, when AR precipitation anomalies were observed. Rutz et al. [213] analyzed spatial distribution of AR duration, AR frequency, and fraction of precipitation from land-falling ARs along the west coast of the United States. Precipitation extremes in the CMIP5 ensemble increase for some regions with global warming, and the return periods of extreme precipitation events become shorter under the higher radiative forcing scenarios by the end of the 21st century, but the pattern of changes in extreme precipitation properties varies with different regions and different models [225, 103]. One might expect that the frequency and intensity of ARs would change with the increasing atmospheric moisture caused by global warming. In this study, our extreme precipitation metric is the AR version of R$\times$1day. This CMIP5 ensemble average of this field over California increases, particularly in the northern and central portion of the state under CMIP5. North American decreases in the indices by [225] are more prominent in the dry subtropics and are especially apparent in Mexico. Along these lines, Dettinger [37] investigated seven GCM models for historical and future climate simulations and showed that AR storms in California become more frequent and more severe under a warming (A2 emission) scenario in the future projections. Lavers et al. [121] recently analyzed the projected AR changes in five CMIP5 GCMs for the RCP8.5 (high emissions) scenario and showed that future Atlantic ARs exhibit increased water vapor with a higher risk of heavy rainfall in Britain.

Climate extremes can be analyzed via a specific statistical theory, the so-called extreme value theory, to quantify the distribution of extreme values (e.g., annual maximum, annual minimum, or excesses over high thresholds) and the probability of the rare events (e.g., [27]). Generalized extreme value (GEV) and peaks-over-threshold (POT) techniques have been successfully applied to extreme meteorological events to analyze the return values of extreme temperature or precipitation events [104, 26, 103, 263]. Statistical methods (in particular point process approaches) based on extreme value analysis have also been applied to the characterization of trends in hot spells and heat waves [50].

In spatial extreme value analysis, multivariate extreme value theory analyzes the dependence of extremes at multiple locations. Max-stable processes (in the framework of stochastic processes) have been applied to model joint distribution of spatial extremes and their dependence [68, 227]. Cooley et al. [30] developed an estimator to measure the degree of spatial dependence between extremes at two locations. In another study by [10], heavy rainfall in France was spatially clustered based on the measurement of spatial dependence. Weller et al. [268] investigated the tail dependence in daily precipitation between reanalysis output and observations of Pineapple Express events using bivariate extreme value theory.

The novel aspects of our study follow from the statistical analysis of extreme precipitation associated with ARs, using a concept of a spatial dependence structure. Here, we quantify spatial extremal dependence to analyze extreme precipitation events caused by ARs associated with large-scale coherent weather systems, focusing on the California region. This study provides detailed characterizations of changes in AR properties and the spatial dependence of extreme rainfall under AR conditions in projections of future climate change. Our principal motivation behind this study is to better understand the characteristics of extreme precipitation in ARs in a warming scenario. As the physical mechanisms causing different types of storms differ, it can be expected that their statistical descriptions will differ as well. In 4.2.3, we introduce the framework of TECA developed at Lawrence Berkeley National Laboratory as an AR identifier, describe our methodology for spatial extreme value analysis, and introduce our metric of spatial dependence. We focus on the characterization of spatial properties in extreme precipitation from ARs identified in ensembles of climate simulations from CMIP5.

### 4.2.3 Methodologies

#### 4.2.3.1 Characteristics of Atmospheric Rivers

ARs play a prominent role in the climatology of inland precipitation over the western United States. Ralph et al. [195] investigated the structure of AR events over the eastern Pacific Ocean, using dropsonde observations. ARs are typically 2,000 or more kilometers long and a few hundred kilometers wide, and the integrated water vapor (IWV) in the atmosphere during AR events is greater than 20 mm, producing heavy rainfall; for example, >250 mm of rain in 60 h for a storm in February 2004 in northern California [197, 154]. Warner et al. [257] used National Climate Data Center (NCDC) daily precipitation observations for 1950-2009 to investigate the top 50 storm events in two-day precipitation at six coastal stations (a total of 207 events in the top 50 for 60 years for multiple stations) along the Pacific Northwest coast. From NCEP-NCAR reanalysis data in [257], most events are associated with ARs, and the extreme precipitation events typically occur in November, December and January, producing heavy rainfall for the periods from 16 to 86 h. Payne and Magnusdottir [177] characterized landfalling ARs along the west coast of North America for the extended winter (November-March) during 1979-2011 using Modern-Era Retrospective Analysis for Research and Applications (MERRA) reanalysis data. The largest number of landfalling AR dates during the season occur in November (6.1 days, averaged for total dates of the month), December (5.8 days), and January (5.2 days). The most intense landfalling dates occur in November (302.3 kg $\mathrm{m}^{-1}$ $\mathrm{s}^{-1}$, peak daily moisture flux), with less intense AR dates later in the extended winter over the western coastline of North America. Rutz et al. [213] indicates that AR-related precipitation is strongly influenced by three factors: climatological characteristics of ARs along the U.S. west coast, water vapor depletion over high mountain barriers, and increase in surface elevation and decrease in pressure. Lavers et al. [120] used atmospheric reanalysis products for the detection of ARs and linked the steep slopes of basins and basin orientation to the high precipitation delivered by the ARs. Lundquist et al. [133] discussed the formation of Sierra Barrier Jets (SBJ) when the atmospheric flow is steered to be roughly parallel to the mountain range.

#### 4.2.3.2 Detection of Atmospheric Rivers

Our AR detection code (described in detail in [13]) utilizes the following scheme for detecting ARs. We first compute a 2D total column integrated precipitable water ($prw$) field by performing a vertical integral on the specific humidity ($hus$) field. The vertical integral of $hus$ was performed over all of the prescribed standard output pressure layers as daily average values of $prw$ were not part of the CMIP5 protocols. The integration was done for the full vertical extent of the modeled atmospheres, but any levels above the tropopause contribute very little to $prw$. The integration is performed on daily data, and the detections are reported on a daily basis as well. We tracked AR only over the northern Pacific Ocean region in order to focus our analysis on ARs that intersect with the California coastline. Following the definition of the physical features of an AR [196, 154], we conduct a thresholding operation for identifying all grid points with $prw > 2cm$, the threshold taken from existing literature. We then use a connected component labeling algorithm to find all of the connected regions of grid points satisfying this criterion for elevated precipitable water. Our implementation of connected component labeling is based on a two-pass algorithm described in [271] and [13]. Finally, we apply geometric constraints to the resulting merged polygons (connected components), which represent a set of candidate atmospheric rivers. These geometric constraints trace $prw$ from a departure point from the tropics (at the west side of the feature) to the intersection with the west coast of North America. Hence, the ARs considered in this study are a subset of all possible ARs across the globe. For all the polygons satisfying these origination and termination conditions, we first construct a medial axis, then compute the length parallel and width perpendicular to this axis, and finally check whether the $length\ of\ the\ AR > 2000km$ and if the $width\ of\ the\ AR < 1000km$. If a polygon satisfies all of these geometric constraints, we identify it as an atmospheric river and note the date of the occurrence. In Figure 4.10, we show a sample AR event detected in the CCSM4 model data (highlighted in the red box) using the AR detection code of TECA. The AR is identified on the simulated date of December 17th, 2005, where the landfall points of the AR are (29.686°N, 125.000°W) and (44.764°N, 126.250°W) in the latitude and longitude coordinates, the width of the AR is 657.713 km, and the length of the AR is 7529.872 km. All further processing considers analysis of precipitation values in California for those specific dates in which ARs are detected.

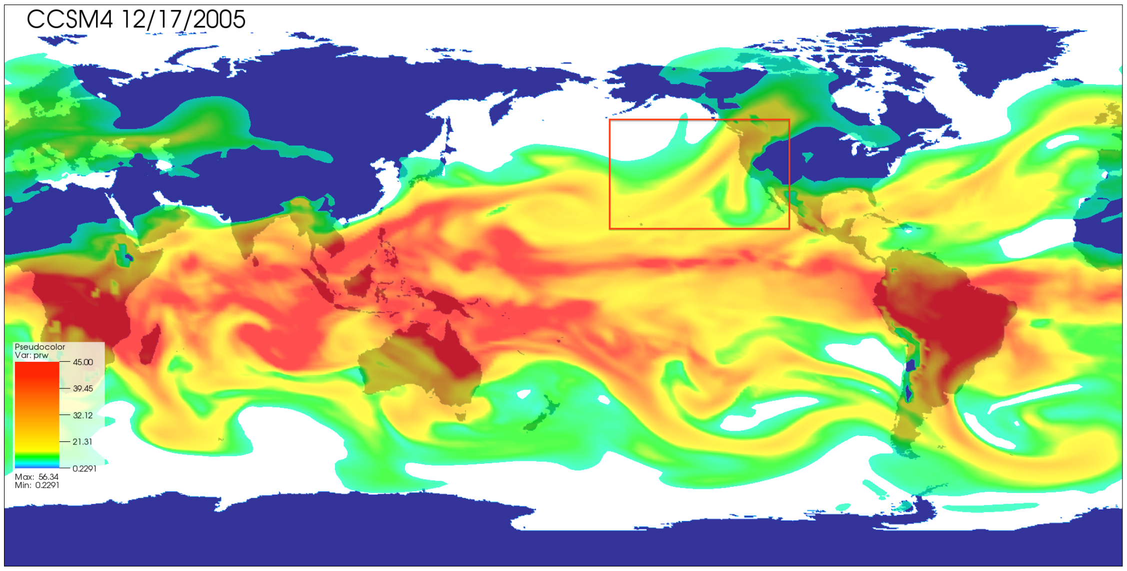


Figure 4.10: Total daily precipitable water ($prw$) field (in $kg/m^2$) from a sample of the CCSM4 model output. It shows a typical AR detected by TECA that originated at the tropical region near Hawaii as it reaches the western coast of the U.S. and identified by TECA detection procedure.

#### 4.2.3.3 Data Description

In this analysis, we consider two variables from CMIP5 models for the historical and representative concentration pathway 8.5 (RCP8.5, the most aggressive warming scenario considered in the CMIP5) experiments: (1) daily specific humidity ($hus$, the mass fraction of water vapor in moist air, in $kg/kg$) to obtain total precipitable water ($prw$) by integrating $hus$ in the vertical dimension over all available vertical model levels for AR detection, and (2) daily precipitation ($pr$, precipitation flux, in $kg\ m^{-2}s^{-1}$) to characterize spatial properties of heavy rainfall under AR events. These two variables were extracted from the Earth System Grid Federation (ESGF) archive for ten CMIP5 multi-models (listed in Table 4.1) with all available ensemble members, which were used to account for the uncertainty of each model experiment. For each model, $pr$ was regridded to a common grid of resolution $1.25° \times 0.94°$ to aid in comparison. The performance of CMIP5 models in simulating extreme precipitation from all types of storms is evaluated in more detail [224, 103].

| Number | Model | Modeling group | Ensemble runs |
|---|---|---|---|
| 1 | CanESM2 | Canadian Centre for Climate Modelling and Analysis | 4 |
| 2 | CCSM4 | National Center for Atmospheric Research | 1 |
| 3 | GFDL-ESM2M | NOAA Geophysical Fluid Dynamics Laboratory | 1 |
| 4 | IPSL-CM5A-LR | Institut Pierre-Simon Laplace | 4 |
| 5 | IPSL-CM5A-MR | Institut Pierre-Simon Laplace | 1 |
| 6 | MIROC5 | AORI (Atmosphere and Ocean Research Institute), NIES (National Institute for Environmental Studies), JAMSTEC (Japan Agency for Marine-Earth Science and Technology), Japan | 2 |
| 7 | MIROC-ESM-CHEM | AORI (Atmosphere and Ocean Research Institute), NIES (National Institute for Environmental Studies), JAMSTEC (Japan Agency for Marine-Earth Science and Technology), Japan | 1 |
| 8 | MPI-ESM-LR | Max Planck Institute for Meteorology | 3 |
| 9 | MRI-CGCM3 | Meteorological Research Institute | 1 |
| 10 | NorESM1-M | Norwegian Climate Centre | 1 |
| Totals | | | 19 |

Table 4.1: The CMIP5 models, modeling groups, and ensemble member(s) used for historical (1981-2005) and RCP8.5 runs (2076-2100) in the study.

Langford et al. [116] examined mean precipitation variability in CMIP5 historical simulations for the California region. The CMIP5 models simulate higher mean precipitation on average than reanalysis datasets, and some models (IPSL-CM5B-LR, MRI-CGCM3, and GISS) show high temporal variability on average in the winter months, when a peak in precipitation typically occurs in the California region. Therefore discrepancies between model and observations or their proxy products would need to be assessed carefully before usage by decision makers.

Our goal in this study is to explore how the frequency and duration of ARs change in future, warmer climates together with the resulting changes in the spatial properties of extreme precipitation events. Changes in the statistics of AR events and extreme precipitations are explored by comparing two 25-year time periods spanning 1981-2005 and 2076-2100. Here, we use the calendar year from January 1 to December 31 and treat an AR event that bridges new year as the event in year when it starts. We focus on the occurrence of ARs in California and note that most AR events in California occur in winter and spring. For selected grid points in California, we take the maximum value from the time series of daily precipitation only for the dates in which ARs are detected within a given year and define this as the *annual maximum AR precipitation*. In order to compare AR-related extreme precipitation against overall extreme precipitation driven by all sources of events, we also compute the annual maximum precipitation by taking the maximum value from daily precipitation during the entire year at each grid point.

#### 4.2.3.4 Spatial Extreme Value Analysis

We use spatial extreme value analysis to quantify the relationships between extreme events occurring at different locations on different days in the same year. Dependence between extremes is affected by the geographic location. Spatial covariance structure is useful for modeling the dependence between spatially distributed variables where the correlations is defined as a function of distance. Under a spatial analysis framework, one can characterize dependence between extremes at two locations by making the modeling assumption regarding the covariance structure (the details of the covariance structure would be described later on). Spatial dependence structure in this study describes whether extremes at different sites, which may occur from distinct ARs during the whole time period, show more or less similarity when the sites are physically close to or far apart from each other.

One appropriate statistical methodology for modeling spatial extremes is the max-stable process approach. Max-stable processes have been used to stochastically model the joint distribution of extreme values at multiple sites [68, 227, 216, 97]. The process works as follows: Let $X(\boldsymbol{s}_i)$ denote a spatial process in which climate variables such as daily precipitation are observed at locations $\boldsymbol{s}_i$ $(i = 1, \cdots, n)$ on a spatial domain. Then, after suitable normalization of $X(\boldsymbol{s})$, the maximum of the normalized process, for example annual or seasonal maximum precipitation, converges in probability distribution to a process $Y(\boldsymbol{s})$:

$$\max_{i=1,\cdots,n} \frac{X(\boldsymbol{s}_i) - b_n(\boldsymbol{s})}{a_n(\boldsymbol{s})} \to Y(\boldsymbol{s}), \tag{4.1}$$

and the limit process $Y(\boldsymbol{s})$ is called a *max-stable process*. In the max-stable modeling approach, spatial dependence can be characterized by a covariance structure between spatial processes at different locations, which is specified by a *variogram*. The variogram is a function of distance between (typically two) spatial locations and is commonly used for modeling the spatial variance and correlation in geo-statistics. Here, we assume an isotropy of spatial process, and the variogram function depends only on the distance of locations. In the modeling procedure, the covariance structure (referred as dependence structure) is modeled by fitting max-stable processes with a parametric form of the variogram function (i.e., a function of parameters), and we estimate the parameters of the variogram to maximize the likelihood of max-stable model for extreme events.

For example, Kabluchko et al. [97] show how a class of max-stable processes known as *Brown-Resnick processes* can be used to model extreme phenomena at multiple locations by the Brownian stochastic process. For the Brown-Resnick process with the dependence structure modeled by a variogram $\gamma$, the bivariate probability distribution is formulated as follows:

$$\begin{aligned} &P\big(Y(s_1) \le y_1, Y(s_2) \le y_2\big) \\ &\quad = \exp\left\{ -\frac{1}{y_1}\Phi\left(\frac{\sqrt{\gamma(h)}}{2} + \frac{1}{\sqrt{\gamma(h)}}\log\frac{y_2}{y_1}\right) - \frac{1}{y_2}\Phi\left(\frac{\sqrt{\gamma(h)}}{2} + \frac{1}{\sqrt{\gamma(h)}}\log\frac{y_1}{y_2}\right)\right\} \end{aligned} \tag{4.2}$$

where $\Phi$ is the standard normal distribution function and $h$ is the Euclidean distance between location $s_1$ and $s_2$. When an exponential variogram, i.e., $\gamma(h) = \theta_0(1 - e^{-h/\theta_1})$, is used in the process of fitting the max-stable model, we parameterize the variogram function with two parameters: (1) $\theta_0$, the sill parameter, describing the variance in the random field; and (2) $\theta_1$, the range parameter, representing the range of dependence.

An *extremal coefficient* provides numerical values in a specific interval for the measurement of dependence between extremes, while the estimation of variogram function provides information on covariance structure of spatial processes for extremes. The extremal coefficient quantifies the degree of spatial dependence for extremes at different locations and is based on the multivariate extreme value theory [217]. The pairwise extremal coefficient, widely used for expressing the dependence between a pair of extremes at two locations, takes values between 1 (complete dependence) and 2 (complete independence). Smith [227] proposed a naive estimator of the pairwise extremal coefficient. Cooley et al. [30] and Naveau et al. [151] investigated a non-parametric approach in characterizing the pairwise dependence among maxima. In this study, we apply the approach by [30] to assess pairwise dependence.

To summarize our methodology, we first identify those AR events via the TECA detection procedure that make landfall in California in the CMIP5 simulations. We then characterize the changes in total AR days and AR frequency by comparing the intensity of extreme precipitation in present-day and future runs conditioned on the occurrence of ARs. We also characterize the spatial patterns of dependence for future projections under climate change within the framework of extreme value theory. For the application of spatial tail dependence, we fit the Brown-Resnick max-stable processes with a power law variogram, $\gamma(h) = \theta_1 h^{\theta_2}$, and quantify the dependence structure by estimating the parameters, $\theta_1$ (range parameter) and $\theta_2$ (smoothing parameter, representing the smoothness of the variogram function). We also estimate the extremal coefficients using the madogram estimator developed by [30] to determine the pairwise dependence of extreme AR precipitation between a focal point and other grid points. The analysis of spatial dependence provides useful information of how extreme values measured at different locations will be correlated spatially. We can also find changes in spatial dependence between extremes induced by AR events in a changing climate, implying changes in spatial coherence of dependence structure between extremes within a warmer climate.

The application of the TECA AR detection capability, followed by spatial analysis results to a climate change scenario were discussed in Section 3.2.3.

## 4.3 Deep Learning for Classification

### 4.3.1 Collaborators

The research in this section was originally published in [129]. The collaborators were **Yunjie Liu**, **Evan Racah**, **Joaquin Correa**, **Amir Khosrowshahi**, **David Lavers**, **Kenneth Kunkel**, **Michael Wehner**, and **William Collins**.

### 4.3.2 Introduction

Extreme climate events (such as hurricanes and heat waves) pose great potential risk on infrastructure and human health. Hurricane Joaquin, for example, hit Carolina in early October 2015 and dropped over 2 feet of precipitation in days, resulting in severe flooding and economic loss. An important scientific goal in climate science research is to characterize extreme events in current day and future climate projections. However, understanding the developing mechanism and life cycle of these events as well as future trend requires accurately identifying such pattern in space and time. Satellites acquire 10s of TBs of global data every year to provide us with insights into the evolution of the climate system. In addition, high resolution climate models produces 100s of TBs of data from multi-decadal run to enable us to explore future climate scenarios under global warming. Detecting extreme climate events in terabytes of data presents an unprecedented challenge for climate science.

Existing extreme climate events (e.g. hurricane) detection methods all build upon human expertise in defining relevant events based on evaluating relevant spatial and temporal variables on hard and subjective thresholds. For instance, tropical cyclones are strong rotating weather systems that are characterized by low pressure, warm temperature core structures with high wind. However, there is no universally accepted sets of criteria for what defines a tropical cyclone [162]. The "Low" Pressure and "Warm" Temperature are interpreted differently among climate scientists; therefore, different thresholds are used to characterize them. Researchers [249, 248, 250, 252, 187, 184] have developed various algorithms to detect tropical cyclones in large climate dataset based on subjective thresholding of several relevant variables (e.g. sea level pressure, temperature, wind etc.). One of the general and promising extreme climate event detecting softwares, Toolkit for Extreme Climate Analysis (TECA) [187, 184], is able to detect tropical cyclones, extra-tropical cyclones, and atmospheric rivers. TECA utilizes the MapReduce paradigm to find pattern in Terabytes of climate data within hours. However, many climate extreme events do not have a clear empirical definition that is universally accepted by climate scientists (e.g. extra-tropical cyclone and mesoscale convective system), which precludes the development and application of algorithms for detection and tracking. This study attempts to search for an alternative methodology for extreme events detection by designing a neural network based system that is capable of learning a broad class of patterns from complex multi-variable climate data and of avoiding subjective threshold.

Recent advances in deep learning have demonstrated exciting and promising results on pattern recognition tasks, such as ImageNet Large Scale Visual Recognition Challenge [111, 226, 238] and speech recognition [78, 33, 62, 235]. Many of the state-of-art deep learning architectures for visual pattern recognition are based on the hierarchical feature learning convolutional neural network (CNN). Modern CNN systems tend to be deep and large with many hidden layers and millions of neurons, making them flexible in learning a broad class of patterns simultaneously from data. AlexNet (7 layers with 5 convolutonal layer and 2 fully connected layer) developed by [111] provides the first end-to-end trainable deep learning system on objective classification, which achieved 15.3% top-5 classification error rate on ILSVRC-2012 data set. On the contrary, previous best performed non-neural network based systems achieved only 25.7% top-5 classification error on the same data set. Shortly after that, Simonyan and Zisserman [226] further developed AlexNet and introduced an even deeper CNN (19 layers with 16 convolutional layer and 3 fully connected layer) with smaller kernel (filter) and achieved an impressively 6.8% top-5 classification error rate on ILSVRC-2014 data set. Szegedy et al.[238] introduced the "inception" neural network concept (network includes sub-network) and developed an even deeper CNN (22 layers) that achieved comparable classification results on ImageNet benchmark. Build on deep CNN, Sermanet et al. [219] introduced an integrated system of classification and detection, in which features learned by convolutional layers are shared among classification and localization tasks and both tasks are performed simultaneously in a single network. Girshick et al. [55] took a completely different approach by combining a region proposal framework [243] with deep CNN and designed the state-of-the-art R-CNN object detection system.

In what follows, we formulate the problem of detecting extreme climate events as classic visual pattern recognition problem. We then build end-to-end trainable deep CNN systems, following the architecture introduced by [111]. The model was trained to classify tropical cyclone, weather front and atmospheric river. Unlike the ImageNet challenge, where the training data are labeled natural images, our training data consist of several continuous spatial variables (e.g. pressure, temperature, precipitation) and are stacked together into image patches.

### 4.3.3 Related Work

Climate data analysis requires an array of advanced methodology. Neural-network-based machine learning approach, as a generative analysis technique, has received much attention and been applied to tackle several climate problems in recent year. Chattopadhyay et al. [18] developed a nonlinear clustering method based on Self Organizational Map (SOM) to study the structure evolution of Madden-Julian oscillation (MJO). Their method does not require selecting leading modes or intraseasonal bandpass filtering in time and space like other methods do. The results show the SOM-based method is not only able to capture the gross feature in MJO structure and development but also reveals insights that other methods are not able to discover such as the dipole and tripole structure of outgoing long wave radiation and diabatic heating in MJO. Gorricha and Costa [60] used a three-dimensional Self Organizational Map on categorizing and visualizing extreme precipitation patterns over an island in Spain. They found spatial precipitation patterns that traditional precipitation index approach is not able to discover and concluded that the three-dimensional Self Organizational Map is a very useful tool on exploratory spatial pattern analysis. More recently, Shi et al. [220] implemented a newly developed convolutional long-short-term memory (LSTM) deep neural network for precipitation nowcasting. Trained on two-dimensional radar map time series, their system is able to outperform the current state-of-the-art precipitation nowcasting system on various evaluation metrics. Iglesias et al. [84] developed a multitask deep fully connected neural network on prediction heat waves trained on historical time series data. They demonstrate that neural network approach is significantly better than linear and logistic regression and potentially can improve the performance of forecasting extreme heat waves. These studies show that neural network is a generative method and can be applied on various climate problems. In this study, we explore deep Convolutional Neural Network on solving climate pattern detection problem.

### 4.3.4 Methods

#### 4.3.4.1 Convolutional Neural Network

A Deep CNN is typically comprised of several convolutional layers followed by a small number of fully connected layers. In between two successive convolutional layers, subsampling operation (e.g. max pooling, mean pooling) is typically performed. Researchers have argued about the necessity of pooling layers and argue that they can be simply replaced by convolutional layer with increased strides, thus simplifying the network structure [229]. In either case, the inputs of a CNN is *(m,n,p)* images, where $m$ and $n$ are the width and height of an image in pixel and $p$ is the number of color channel of each pixel. The output of a CNN is a vector of $q$ probability units (class scores), corresponding to the number of categories to be classified (e.g. for binary classifier $q$=2).

The convolutional layers perform convolution operation between kernels and the input images (or feature maps from previous layer). Typically, a convolutional layer contains $k$ filters (kernels) with the size *(i,j,p)*, where *i, j* are the width and height of the filter. The filters are usually smaller than the width $m$ and height $n$ of input image, and $p$ is always equal to the number of color channel of input image (e.g. a color image has three channels: red, green, and blue). Each of the filters is independently convolved with the input images (or feature maps from previous layer) followed by non-linear transformation and generates $k$ feature maps, which serve as inputs for the next layer. In the process of convolution, a dot product is computed between the entry of filter and the local region that is connected to the input image (or feature map from previous layer). The parameters of convolutional layer are these learnable filters. The convolutional layer is the feature extractor, because the kernels slide across all the inputs and will produce larger outputs for certain sub-regions than for others. This allows features to be extracted from inputs and preserved in the feature maps, which are passed on to next layer, regardless of where the feature is located in the input. The pooling layer subsamples the feature maps generated from convolutional layer over a *(s,t)* contiguous region, where *s, t* are the width and height of the subsampling window. This results in the resolution of the feature maps becoming coarser with the depth of CNN. All feature maps are high-level representations of the input data in CNN. The fully connected layer has connections to all hidden units in previous layer. If it is the last layer within the CNN architecture, the fully connected layer also does the high level reasoning based on the feature vectors from previous layer and produce final class scores for image objects.

Most current deep neural networks use back propagation as learning rule [209]. The back propagation algorithm searches for the minimum of the loss function in the weight space through gradient descent method. It partitions the final total loss to each of the single neuron in the network and repeatedly adjusts the weights of neurons whose loss is high, and it back propagates the error through the entire network from output to its inputs.

#### 4.3.4.2 Hyper-parameter Optimization

Training deep neural network is known to be hard [117, 56]. Effectively and efficiently training a deep neural network not only requires a large amount of training data but also requires carefully tuning model hyper-parameters (e.g. learning parameters, regularization parameters) [228]. The parameter tuning process, however, can be tedious and non-intuitive. Hyper-parameter optimization can be reduced to find a set of parameters for a network that produces the best possible validation performance. As such, this process can be thought of as a typical optimization problem of finding a set, $x$, of parameter values from a bounded set $X$ that minimize an objective function $f(x)$, where $x$ is a particular setting of the hyper-parameters and $f(x)$ is the loss for a deep neural network with a particular set of training and testing data as function of the hyper-parameter inputs. Training a deep neural network is not only a costly (with respect to time) procedure but a rather opaque process with respect to how the network performance varies with respect to its hyper-parameter inputs. Because training and validating a deep neural network is very complicated and expensive, Bayesian Optimization (which assumes $f(x)$ is not known, is non-convex and is expensive to evaluate) is a well-suited algorithm for hyper-parameter optimization for our task at hand. Bayesian Optimization attempts to optimize $f(x)$ by constructing two things: a probabilistic model of $f(x)$ and an acquistion function that picks which point $x$ in $X$ to evaluate next. The probabilistic model is updated with Baye's rule with a Gaussian prior. The acquisition function suggests hyper-parameter settings or points to evaluate by trying to balance evaluating parameter settings in regions, where $f(x)$ is low and points in regions where the uncertainty in the probabilistic model is high. As a result, the optimization procedure attempts to evaluate as few points as possible [12] [228].

In order to implement Bayesian Optimization, we use a tool called Spearmint. Spearmint works by launching a Spearmint master process, which creates a database for collecting all model evaluation results. The master process then spawns many processes, which execute training and evaluation with respect to a set of hyper-parameters proposed by the acquisition function and then report their results to the database. From there, the master process uses the results in the database to propose further parameter settings and launch additional processes.

#### 4.3.4.3 CNN Configuration

Following AlexNet [111], we developed a deep CNN which has 4 learnable layers, including 2 convolutional layers and 2 fully connected layers. Each convolutional layer is followed by a

| | Conv1 | Pooling | Conv2 | Pooling | Fully | Fully |
|---|---|---|---|---|---|---|
| Tropical Cyclone | 5x5-8 | 2x2 | 5x5-16 | 2x2 | 50 | 2 |
| Weather Fronts | 5x5-8 | 2x2 | 5x5-16 | 2x2 | 50 | 2 |
| Atmospheric River | 12x12-8 | 3x3 | 12x12-16 | 2x2 | 200 | 2 |

Table 4.2: Deep CNN architecture and layer parameters. The convolutional layer parameters are denoted as <filter size>-<number of feature maps> (e.g. 5x5-8). The pooling layer parameters are denoted as <pooling window> (e.g. 2x2). The fully connected layer parameter are denoted as <number of units> (e.g. 2).

max pooling layer. The model is constructed based on the open-source Python deep learning library NOEN. The configuration of our best performed architectures are shown in Table 4.2.

The networks are shallower and smaller comparing to the state-of-the-art architecture developed by [226, 238]. The major limitations for exploring deeper and larger CNNs are the limited amount of labeled training data that we can obtain. However, a small network has the advantage of avoiding over-fitting, especially when the amount of training data is small. We also chose comparatively large kernels (filters) in the convolutional layer based on input data size, even though [226] suggests that deep architecture with small kernel (filter) is essential for state-of-the-art performance. This is because climate patterns are comparatively simpler and larger in size as compared to objects in ImageNet dataset.

One key feature of deep learning architectures is that it is able to learn complex non-linear functions. The convolutional layers and first fully connected layer in our deep CNNs all have Rectified Linear Unit (ReLU) activation functions [149] as characteristic. ReLU is chosen due to its faster learning/training character [111] as compared to other activation functions like tanh.

$$f(x) = max(0, x) \tag{4.3}$$

The final fully connected layer has Logistic activation function as non-linearity, which also serves as classifier and outputs a probability distribution over class labels.

$$f(x) = \frac{1}{1 + e^{-x}} \tag{4.4}$$

### 4.3.4.4 Computational Platform

We performed our data processing, model training and testing on Edison, a Cray XC30 and Cori, a Cray XC40 supercomputing systems at the National Energy Research Scientific Computing Center (NERSC). Each of Edison computing nodes has 24 2.4 GHz Intel Xeon processors. Each of Cori computing nodes has 32 2.3 GHz Intel Haswell processors. In our work, we mainly used single node CPU backend of NEON. The hyper-parameter optimization was performed on a single node on Cori with tasks fully parallel on 32 cores.

| Climate Dataset | Time Frame | Temporal Resolution | Spatial Resolution (lat x lon degree) |
|---|---|---|---|
| CAM5.1 historical run | 1979-2005 | 3 hourly | 0.23x0.31 |
| ERA-Interim reanalysis | 1979-2011 | 3 hourly | 0.25x0.25 |
| 20 century reanalysis | 1908-1948 | Daily | 1x1 |
| NCEP-NCAR reanalysis | 1949-2009 | Daily | 1x1 |

Table 4.3: Data Sources used in binary classification study.

| Events | Image Dimension | Variables | Total Examples |
|---|---|---|---|
| Tropical Cyclone | 32x32 | PSL,VBOT,UBOT, T200,T500,TMQ, V850,U850 | 10,000 +ve 10,000 -ve |
| Atmospheric River | 148 x 224 | TMQ,Land Sea Mask | 6,500 +ve 6,800 -ve |
| Weather Front | 27 x 60 | 2m Temp, Precip, SLP | 5,600 +ve 6,500 -ve |

Table 4.4: Size of image patch, diagnostic variables and number of labeled dataset used for extreme event considered in the study

### 4.3.5 Data

In this study, we use both climate simulations and reanalysis products. The reanalysis products are produced by assimilating observations into a climate model. The spatial scale of both climate model simulation and reanalysis products covers the entire global. A summary of the data source and its temporal and spatial resolution is listed in Table 4.3. Ground truth labeling of various events is obtained via multivariate threshold based criteria implemented in TECA [187, 184] and manual labeling by experts [112, 122]. Training data comprise of image patterns, where several relevant spatial variables are stacked together over a prescribed region that bounds a type of event. The dimension of the bounding box is based on domain knowledge of events spatial extent in real word. For instance, tropical cyclone radius are typically within range of 100 kilometers to 500 kilometers, thus bounding box size of 500 kilometers by 500 kilometers is likely to capture most of tropical cyclones. The chosen physical variables are also based on domain expertise. The prescribed bounding box is placed over the event. Relevant variables are extracted within the bounding box from the climate model simulations or reanalysis products and stacked together. To facilitate model training, bounding box location is adjusted slightly such that all of events are located approximately at the center. Image patches are cropped and centered correspondingly. Because the spatial dimension of climate events vary quite a lot and the spatial resolution of source data is non-uniform, final training images prepared differ in their size among the three types of event. The class labels of images are "containing events" and "not containing events"; in other words, we formulate the problem as a binary classification task. A summary of the attributes of training images is listed in Table 4.4.

### 4.3.6 Results and Discussion

Table 4.5 summarizes the performance of our deep CNN architecture on classifying tropical cyclones, atmospheric rivers, and weather fronts. We obtained fairly high accuracy (89%-99%) on extreme event classification. In addition, the systems do not suffer from over-fitting. We believe this is mostly because of the shallow and small size of the architecture (4 learnable layers) and the weight decay regularization. Deeper and larger architecture would be inappropriate for this study due to the limited amount of training data. Fairly good train and test classification results also suggest that the deep CNNs we developed are able to efficiently learn representations of climate pattern from labeled data and make predictions based on feature learned. Traditional threshold-based detection method requires human experts to carefully examine the extreme event and its environment, thus come up with thresholds for defining the events. In contrast, as shown in this study, deep CNNs are able to learn climate pattern just from the labeled data, thus avoiding subjective thresholds.

| Event Type | Train | Test | Train time |
|---|---|---|---|
| Tropical Cyclone | 99% | 99% | ≈ 30 min |
| Atmospheric River | 90.5% | 90% | 6-7 hour |
| Weather Front | 88.7% | 89.4% | ≈ 30 min |

Table 4.5: Overall Classification Accuracy

#### 4.3.6.1 Classification Results for Tropical Cyclones

Tropical cyclones are rapid rotating weather systems that are characterized by low pressure center with strong wind circulating the center and warm temperature core in upper troposphere. Figure 4.11 shows examples of tropical cyclones simulated in climate models, which are correctly classified by deep CNN (warm core structure is not shown in this figure). Tropical cyclone features are rather well defined, as can be seen from the distinct low pressure center and spiral flow of wind vectors around the center. These clear and distinct characteristics make tropical cyclone pattern relatively easy to learn and represent within CNN. Our deep CNNs achieved nearly perfect (99%) classification accuracy.

Figure 4.12 shows examples of tropical cyclones that are mis-classified. After carefully examining these events, we believe they are weak systems (e.g. tropical depression), whose low pressure center and spiral structure of wind have not fully developed. The pressure distribution shows a large low pressure area without a clear minimum. Therefore, our deep CNN does not label them as strong tropical cyclones.

| | Label TC | Label Non_TC |
|---|---|---|
| Predict TC | 0.989 | 0.003 |
| Predict Non_TC | 0.011 | 0.997 |

Table 4.6: Confusion matrix for tropical cyclone binary classification

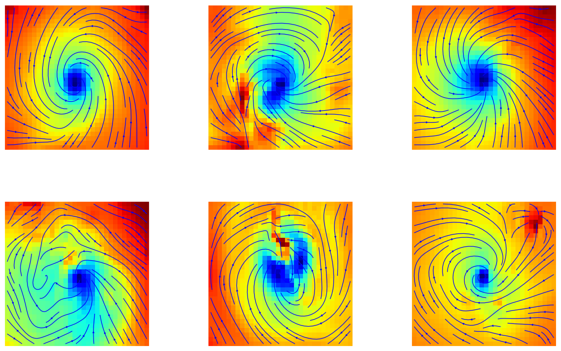


Figure 4.11: Sample images of tropical cyclones correctly classified (true positive) by our deep CNN model. Figure shows sea level pressure (color map) and near surface wind distribution (vector solid line).

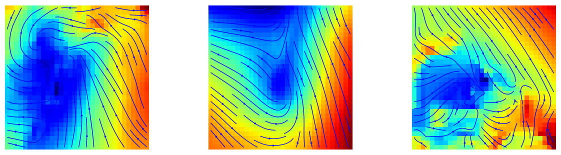


Figure 4.12: Sample images of tropical cyclones mis-classified (false negative) by our deep CNN model. Figure shows sea level pressure (color map) and near surface wind distribution (vector solid line).

#### 4.3.6.2 Classification Results for Atmospheric Rivers

In contrast to tropical cyclones, atmospheric rivers are distinctively different events. They are narrow corridors of concentrated moisture in atmosphere. They usually originate in tropical oceans and move pole-ward. Figure 4.13 shows examples of correctly classified land falling atmospheric rivers that occur on the western Pacific Ocean and north Atlantic Ocean. The characteristics of narrow water vapor corridor is well defined and clearly observable in these images.

Figure 4.14 shows mis-classified atmospheric rivers. Upon further investigation, we believe there are two main factors leading to mis-classification. The first factor is the presence of weak atmospheric river systems. For instance, the left column of Figure 4.14 shows comparatively weak atmospheric rivers. The water vapor distribution clearly shows a band of concentrated moisture cross mid-latitude ocean, but the signal is much weaker comparing to Figure 4.13. Thus, deep CNN does not predict them correctly. The second factor is the presence of other climate events that may also affect deep CNN representation of atmospheric rivers. In reality, the location and shape of atmospheric river are affected by jet streams and extra-tropical cyclones. For example, Figure 4.14 right column shows rotating systems (likely extra-tropical cyclone) adjacent to the atmospheric river. This phenomenon presents challenge for deep CNN on representing atmospheric river.

| | Label AR | Label Non_AR |
|---|---|---|
| Predict AR | 0.93 | 0.107 |
| Predict Non_AR | 0.07 | 0.893 |

Table 4.7: Confusion matrix for atmospheric river binary classification

#### 4.3.6.3 Classification Results for Weather Fronts

Among the three types of climate events we are looking at, weather fronts have the most complex spatial pattern. Weather fronts typically form at the interface of warm air and cold air, and usually associated with heavy precipitation due to moisture condensation of warm air up-lifting. In satellite images, a weather front is observable as a strip of clouds, but it is hardly visible on two dimensional fields such as temperature and pressure. In middle latitude (e.g. most U.S.), a number of weather fronts are associated with extra-tropical cyclones. Figure 4.15 shows examples of correctly classified weather fronts by our deep CNN system. Visually, the narrow long regions of high precipitation line up approximately parallel to the temperature contour. This is a clear characteristics and comparatively easy for deep CNNs to learn.

Because patterns of weather fronts are rather complex and hardly show up in two-dimensional fields, we decided to further investigate them in later work.

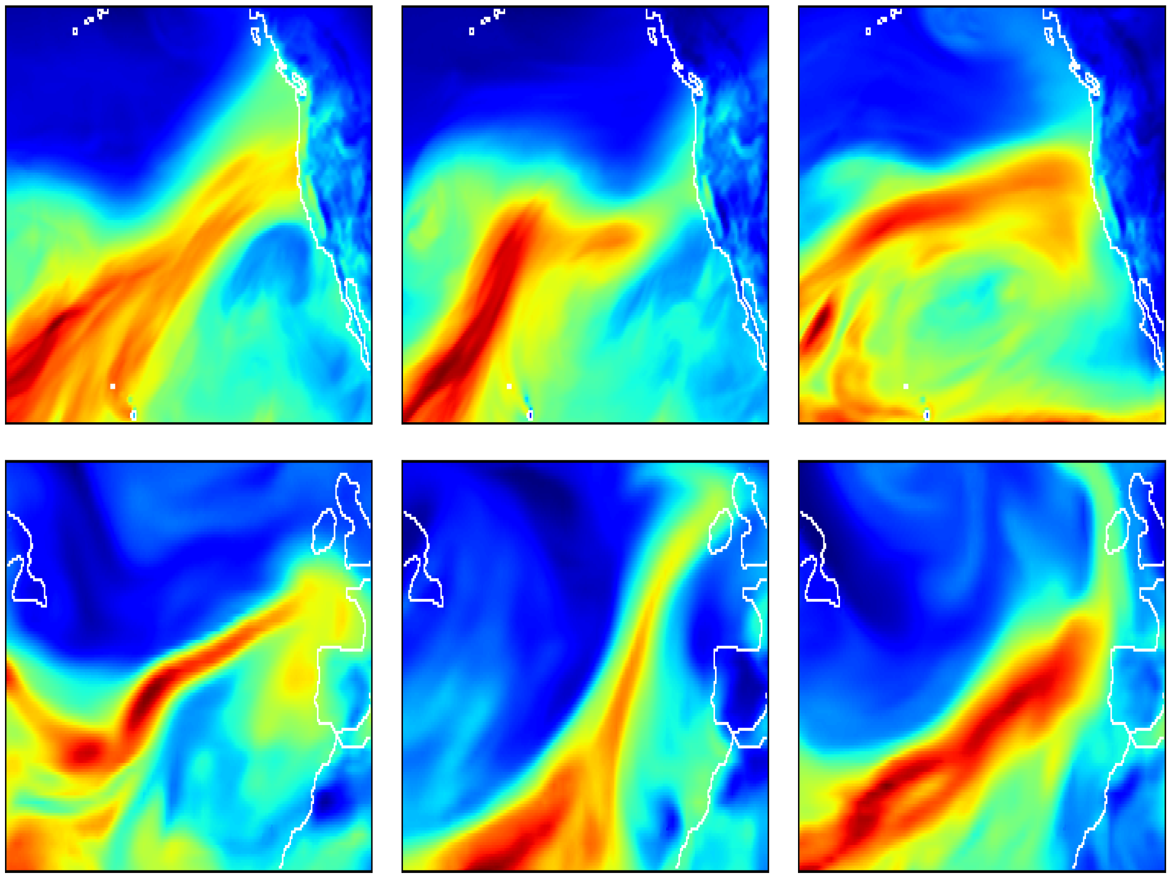


Figure 4.13: Sample images of atmospheric rivers correctly classified (true positive) by our deep CNN model. Figure shows total column water vapor (color map) and land sea boundary (solid line).

| | Label WF | Label Non_WF |
|---|---|---|
| Predict WF | 0.876 | 0.18 |
| Predict Non_WF | 0.124 | 0.82 |

Table 4.8: Confusion matrix for weather front binary classification

### 4.3.7 Future Work

In the present study, we trained deep CNNs separately for classifying tropical cyclones, atmospheric rivers, and weather fronts individually. Ideally, we would like to train a **single** neural network for detecting all three types of events. Unlike object recognition in natural images, climate patterns detection have unique challenges. Firstly, climate events happen at

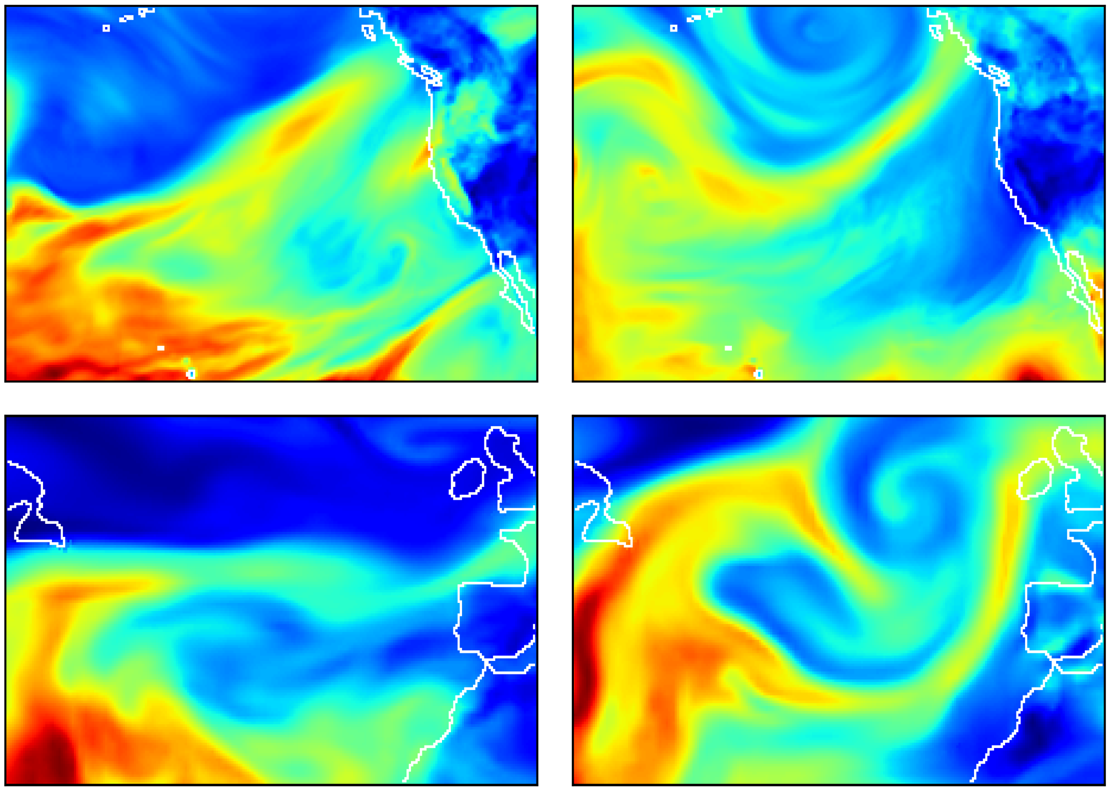


Figure 4.14: Sample images of atmospheric rivers mis-classified (false negative) by our deep CNN model. Figure shows total column water vapor (color map) and land sea boundary (solid line).

vastly different spatial scales. For example, a tropical cyclone typically extends over less than 500 kilometers in radius, while an atmospheric river can be several thousand kilometers long. Secondly, different climate events are characterized by different sets of physical variables. For example, atmospheric rivers correlate strongly with the vertical integration of water vapor, while tropical cyclones have a more complex multi-variable pattern involving sea-level pressure, near-surface wind, and upper-troposphere temperature. Future work will need to develop generative CNN architectures that are capable of discriminating between different variables based on the event type and capable of handling events at various spatial scale. Note that we have primarily addressed **detection** of extreme weather patterns, but not their **localization**. We will consider architectures for spatially localizing weather patterns in the future.

Several researchers have pointed out that deeper and larger CNNs perform better for classification and detection tasks [226, 238] compared to shallow networks. However, deep networks require huge amount of data to be effectively trained and to prevent model over

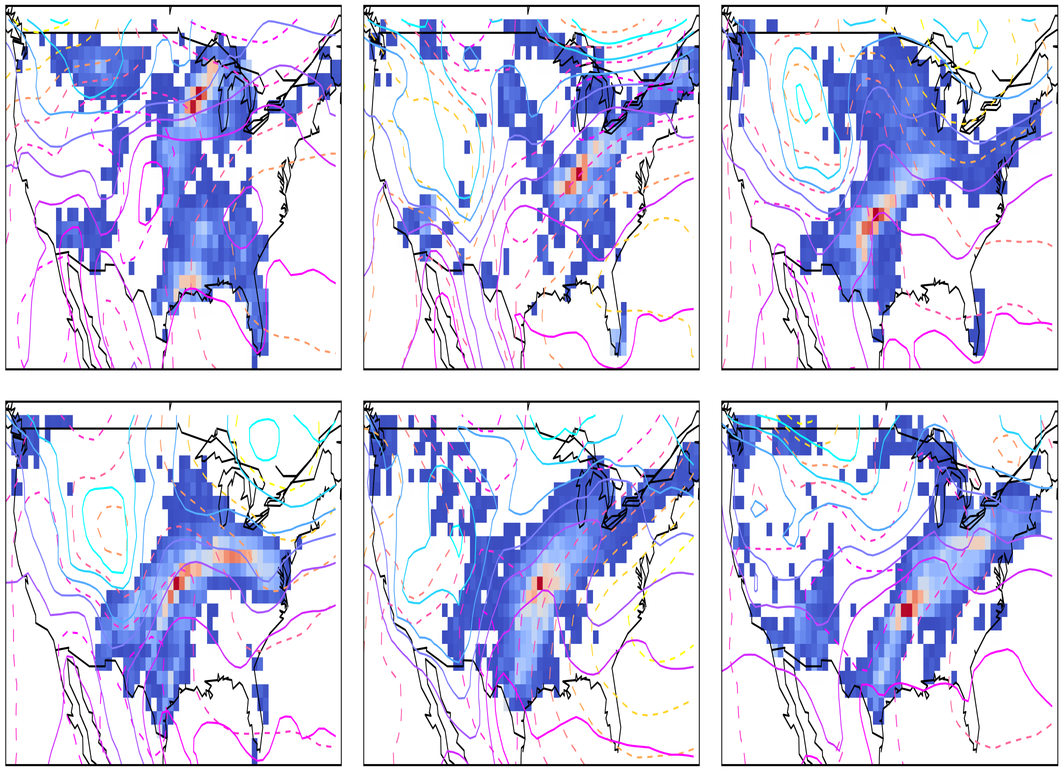


Figure 4.15: Sample images of weather front correctly classified by our deep CNN model. Figure shows precipitation with daily precipitation less than 5 millimeters filtered out (color map), near surface air temperature (solid contour line) and sea level pressure (dashed contour line)

fitting. Datasets, such as ImageNet, provide millions of labeled images for training and testing deep and large CNNs. In contrast, we can only obtain a small amount of labeled training data; hence we are constrained on the class of deep CNNs that we can explore without suffering from over-fitting. This limitation also points us to the need for developing unsupervised approaches for climate pattern detection. We believe that this will be critical for the majority of scientific disciplines that typically lack labeled data.

### 4.3.8 Conclusion

In this study, we explored deep learning as a methodology for detecting extreme weather patterns in climate data. We developed deep CNN architecture for classifying tropical cyclones, atmospheric rivers, and weather fronts. The system achieves fairly high classification accuracy, ranging from 89% to 99%. To the best of our knowledge, this is the first time that deep CNN has been applied to tackle climate pattern recognition problems. This successful application could be a precursor for tackling a broad class of pattern detection problems in climate science. Deep neural network learns high-level representations from data directly, therefore potentially avoiding traditional subjective thresholding based criteria of climate variables for event detection. Results from this study will be used for quantifying climate extreme event trends in current days and future climate scenarios, as well as for investigating the changes in dynamics and thermodynamics of extreme events in global warming contention. This information is critical for climate change adaptation, hazard risk prediction, and climate change policy making.

## 4.4 Deep Learning for Detection

### 4.4.1 Collaborators

The research in this section was originally published in [190]. The collaborators were **Evan Racah**, **Christopher Beckham**, **Tegan Maharaj**, **Samira Ebrahimi Kahou**, and **Christopher Pal**.

### 4.4.2 Introduction

Deep neural networks, especially deep convolutional neural networks, have enjoyed breakthrough success in recent recent years, achieving state-of-the-art results on many benchmark datasets [111, 75, 238] and also compelling results on many practical tasks such as disease diagnosis [81], facial recognition [175], autonomous driving [20], and many others. Furthermore, deep neural networks have also been very effective in the context of unsupervised and semi-supervised learning; some recent examples include variational autoencoders [107], adversarial networks [59, 137, 214, 230], ladder networks [200], and stacked what-where autoencoders [281].

There is a recent trend towards video datasets aimed at better understanding spatiotemporal relations and multimodal inputs [100, 65, 61]. The task of finding extreme weather events in climate data is similar to the task of detecting objects and activities in video - a popular application for deep learning techniques. An important difference is that in the case of climate data, the 'video' has 16 or more 'channels' of information (such as water vapour, pressure and temperature), while conventional video only has 3 (RGB). In addition, climate simulations do not share the same statistics as natural images. As a result, unlike many popular techniques for video, we hypothesize that we cannot build off successes from the computer vision community such as using pretrained weights from CNNs [226, 111] pretrained on ImageNet [210].

Climate data thus poses a number of interesting machine learning problems: multi-class classification with unbalanced classes; partial annotation; anomaly detection; distributional shift and bias correction; spatial, temporal, and spatiotemporal relationships at widely varying scales; relationships between variables that are not fully understood; issues of data and computational efficiency; opportunities for semi-supervised and generative models; and more. Here, we address multi-class detection and localization of four extreme weather phenomena: tropical cyclones, extra-tropical cyclones, tropical depressions, and atmospheric rivers. We implement a 3D (height, width, time) convolutional encoder-decoder, with a novel single-pass bounding-box regression loss applied at the bottleneck. To our knowledge, this is the first use of a deep autoencoding architecture for bounding-box regression. This architectural choice allows us to do semi-supervised learning in a very natural way (simply training the autoencoder with reconstruction for unlabelled data), while providing relatively interpretable features at the bottleneck. This is appealing for use in the climate community, as current engineered heuristics do not perform as well as human experts for identifying extreme weather events.

Our main contributions are (1) a baseline bounding-box loss formulation; (2) our architecture, a first step away from engineered heuristics for extreme weather events, towards semi-supervised learned features; (3) the ExtremeWeather dataset, which we make available in three benchmarking splits: one small, for model exploration, one medium, and one comprising the full 27 years of climate simulation output.

### 4.4.3 Related work

#### 4.4.3.1 Deep learning for climate and weather data

Climate scientists do use basic machine learning techniques, for example PCA analysis for dimensionality reduction [147] and k-means analysis for clusterings [233]. However, the climate science community primarily relies on expert engineered systems and ad-hoc rules for characterizing climate and weather patterns. Of particular relevance is the TECA (Toolkit for Extreme Climate Analysis) [187, 184], an application of large scale pattern detection on climate data using heuristic methods. A more detailed explanation of how TECA works is described in 4.4.4. Using the output of TECA analysis (centers of storms and bounding boxes around these centers) as ground truth, [129] demonstrated for the first time that convolutional architectures could be successfully applied to predict the class label for two extreme weather event types. Their work considered the binary classification task on centered, cropped patches from 2D (single-timestep) multi-channel images. Like [129] we use TECA's output (centers and bounding boxes) as ground truth, but we build on the work of [129] by: 1) using uncropped images, 2) considering the temporal axis of the data, 3) doing multi-class bounding box detection, and 4) taking a semi-supervised approach with a hybrid predictive and reconstructive model.

Some recent work has applied deep learning methods to weather forecasting. Shi Xingjian et al. [274] have explored a convolutional LSTM architecture (described in 4.4.3.2 for predicting future precipitation on a local scale (i.e. the size of a city), using radar echo data. In contrast, we focus on extreme event detection on planetary-scale data. Our aim is to capture patterns which are very local in time (e.g. a hurricane may be present in half a dozen sequential frames), compared to the scale of our underlying climate data, consisting of global simulations over many years. As such, 3D CNNs seemed to make more sense for our detection application, compared to LSTMs whose strength is in capturing long-term dependencies.

#### 4.4.3.2 Related methods and models

Following the dramatic success of CNNs in static 2D images, a wide variety of CNN architectures have been explored for video, ex. [98, 275, 241]. The details of how CNNs are extended to capture the temporal dimension are important. Andrej Karpathy et al. [98] explore different strategies for fusing information from 2D CNN subcomponents; in contrast, Li Yao et al. [275] create 3D volumes of statistics from low level image features.

Convolutional networks have also been combined with RNNs (recurrent neural networks) for modeling video and other sequence data, and we briefly review some relevant video models here. The most common and straightforward approach to modeling sequential images is to feed single-frame representations from a CNN at each timestep to an RNN. This approach has been examined for a number of different types of video [42, 45], while Nitish Srivastava et al. [231] have explored an LSTM architecture for the unsupervised learning of video representations, using a pretrained CNN representation as input. These architectures separate learning of spatial and temporal features, something which is not desirable for climate patterns. Another popular model, also used on 1D data, is a convolutional RNN, wherein the hidden-to-hidden transition layer is 1D convolutional (i.e. the state is convolved over time). Nicolas Ballas et al. [7] combine these ideas, applying a convolutional RNN to frames processed by a (2D) CNN.

The 3D CNNs we use here are based on 3-dimensional convolutional filters, taking the height, width, and time axes into account for each feature map, as opposed to aggregated 2D CNNs. This approach was studied in detail in [241]. 3D convolutional neural networks have been used for various tasks ranging from human activity recognition [95] to large-scale YouTube video classification [98] and video description [275]. Ehsan Hosseini-Asl et al. [81] use a 3D convolutional autoencoder for diagnosing Alzheimer's disease through MRI - in their case, the 3 dimensions are height, width, and depth. William F. Whitney et al. [269] use 3D (height, width, depth) filters to predict consecutive frames of a video game for continuation learning. Recent work has also examined ways to use CNNs to generate animated textures and sounds [273]. This work is similar to our approach in using 3D convolutional encoder, but where their approach is stochastic and used for generation, ours is deterministic, used for multi-class detection and localization, and also comprises a 3D convolutional decoder for unsupervised learning.

Stepping back, our approach is related conceptually to [144], who use semi-supervised learning for bounding-box detection, but their approach uses iterative heuristics with a support vector machine (SVM) classifer, an approach which would not allow learning of spatiotemporal features. Our setup is also similar to recent work from [280] (and others) in using a hybrid prediction and autoencoder loss. This strategy has not, to our knowledge, been applied either to multidimensional data or bounding-box prediction, as we do here.

Our bounding-box prediction loss is inspired by [201], an approach extended in [206], as well as the single shot multiBox detector formulation used in [128] and the seminal bounding-box work in OverFeat [219]. Details of this loss are described in 4.4.5.

### 4.4.4 The ExtremeWeather dataset

#### 4.4.4.1 The Data

The climate science community uses three flavors of global datasets: observational products (satellite, gridded weather station); reanalysis products (obtained by assimilating disparate observational products into a climate model), and simulation products. In this study, we analyze output from the third category because we are interested in climate change projection studies. We would like to better understand how Earth's climate will change by the year 2100, and it is only possible to conduct such an analysis on simulation output. Although this dataset contains the past, the performance of deep learning methods on this dataset can still inform the effectiveness of these approaches on future simulations. We consider the CAM5 (Community Atmospheric Model v5) simulation, which is a standardized three-dimensional, physical model of the atmosphere used by the climate community to simulate the global climate [29]. When it is configured at 25-km spatial resolution [266], each snapshot of the global atmospheric state in the CAM5 model output is a 768x1152 image, having 16 'channels', each corresponding to a different simulated variable (like surface temperature, surface pressure, precipitation, zonal wind, meridional wind, humidity, cloud fraction, water vapor, etc.). The global climate is simulated at a temporal resolution of 3 hours, giving 8 snapshots (images) per day. The data we provide is from a simulation of 27 years from 1979 to 2005. In total, this gives 78,840 16-channel 768x1152 images.

#### 4.4.4.2 The Labels

Ground-truth labels are created for four extreme weather events: Tropical Depressions (TD), Tropical Cyclones (TC), Extra-Tropical Cyclones (ETC), and Atmospheric Rivers (AR), using TECA [187]. TECA generally works by suggesting candidate coordinates for storm centers and by only selecting points that follow a certain combination of criteria, which usually involves requiring various variables, such as pressure, temperature, and wind speed, whose values are between between certain thresholds. These candidates are then refined by breaking ties and matching the "same" storms across time [187]. These storm centers are then used as the center coordinates for bounding boxes. The size of the boxes is determined using prior domain knowledge as to how big these storms usually are, as described in [129]. Every other image (i.e. 4 per day) is labeled due to certain design decisions made during the production run of the TECA code. This gives us 39,420 labeled images.

**4.4.4.2.1 Issues with the Labels** TECA, the ground truth labeling framework, implements heuristics to assign 'ground truth' labels for the four types of extreme weather events. However, it is entirely possible there are errors in the labeling: for instance, there is little agreement in the climate community on a standard heuristic for capturing Extra-Tropical Cyclones [156]; Atmospheric Rivers have been extensively studied in the northern hemisphere [122, 39], but not in the southern hemisphere; and spatial extents of such events are not universally agreed upon. In addition, this labeling only includes AR's in the US and not in Europe. As such, there is potential for many false negatives, resulting in partially annotated images. Lastly, it is worth mentioning that because the ground truth generation is a simple automated method, a deep, supervised method can only do as well as emulating this class of simple functions. This, in addition to lower representation for some classes (AR and TD), is part of our motivation in exploring semi-supervised methods to better understand the features underlying extreme weather events rather than trying to "beat" existing techniques.

#### 4.4.4.3 Suggested Train/Test Splits

We provide suggested train/test splits for the varying sizes of datasets on which we run experiments. Table 4.9 shows the years used for train and test for each dataset size. We show "small" (2 years train, 1 year of test), "medium" (8 years train, 2 years test), and "large" (22 years train, 5 years test) datasets. For reference, table 4.10 shows the breakdown of the dataset splits for each class for "small" in order to illustrate the class-imbalance present in the dataset. Our model was trained on "small", where we split the train set 50:50 for train and validation. Links for downloading train and test data, as well as further information about the different dataset sizes and splits, can be found here: `extremeweatherdataset.github.io`.

| Level | Train | Test |
|---|---|---|
| Small | 1979, 1981 | 1984 |
| Medium | 1979-1983,1989-1991 | 1984-1985 |
| Large | 1979-1983, 1994-2005, 1989-1993 | 1984-1988 |

Table 4.9: Three benchmarking levels for the ExtremeWeather dataset

### 4.4.5 The model

We use a 3D convolutional encoder-decoder architecture, meaning that the filters of the convolutional encoder and decoder are 3 dimensional (height, width, time). The architecture is shown in Figure 4.16; the encoder uses convolution at each layer while the decoder is the equivalent structure in reverse, using tied weights and deconvolutional layers, with leaky ReLUs [6] (0.1) after each layer. As we take a semi-supervised approach, the code (bottleneck) layer of the autoencoder is used as the input to the loss layers, which make predictions for (1) bounding box location and size, (2) class associated with the bounding box, and (3) the

| **Benchmark** | **Split** | **TC** (%) | **ETC** (%) | **TD** (%) | **US-AR** (%) | **Total** |
|---|---|---|---|---|---|---|
| Small | Train | 3190 (42.32) | 3510 (46.57) | 433 (5.74) | 404 (5.36) | 7537 |
| | Test | 2882 (39.04) | 3430 (46.47) | 697 (9.44) | 372 (5.04) | 7381 |

Table 4.10: Number of examples in ExtremeWeather benchmark splits, with class breakdown statistics for Tropical Cyclones (**TC**), Extra-Tropical Cyclones (**ETC**), Tropical Depressions (**TD**), and United States Atmospheric Rivers (**US-AR**)

confidence (sometimes called 'objectness') of the bounding box. Further details (filter size, stride length, padding, output sizes, etc.) can be found in the supplementary materials.

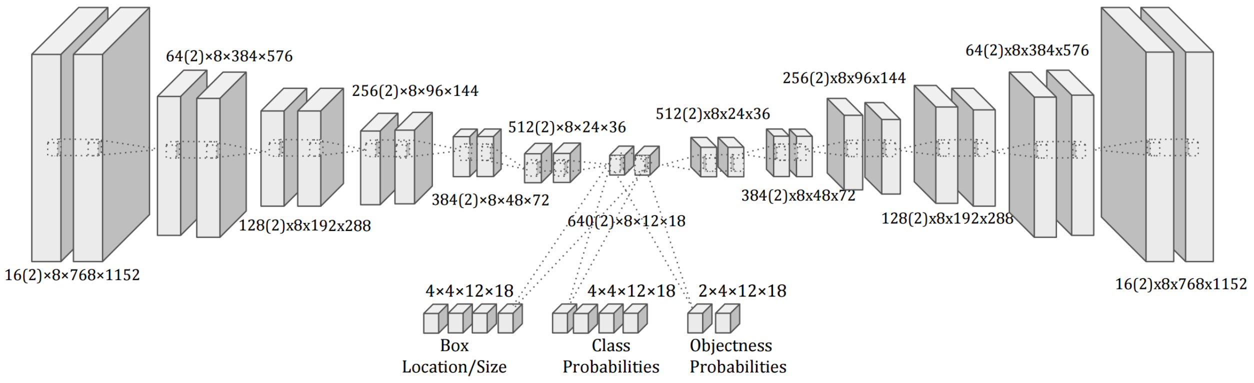


Figure 4.16: Diagram of the 3D semi-supervised architecture. Parentheses denote subset of total dimension shown (for ease of visualization, only two feature maps per layer are shown for the encoder-decoder. All feature maps are shown for bounding-box regression layers).

The total loss for the network, $L$, is a weighted combination of supervised bounding-box regression loss, $L_{sup}$, and unsupervised reconstruction error, $L_{rec}$ :

$$L = L_{sup} + \lambda L_{rec}, \tag{4.5}$$

where $L_{rec}$ is the mean squared squared difference between input $X$ and reconstruction $X^*$:

$$L_{rec} = \frac{1}{M}||X - X^*||_2^2, \tag{4.6}$$

where $M$ is the total number of pixels in an image.

In order to regress bounding boxes, we split the original 768x1152 image into a 12x18 grid of 64x64 anchor boxes. We then predict a box at each grid point by transforming the representation to 12x18=216 scores (one per anchor box). Each score encodes three pieces of information: (1) how much the predicted box differs in size and location from the anchor box, (2) the confidence that an object of interest is in the predicted box ("objectness"), and (3) the class probability distribution for that object. Each component of the score is computed by several 3x3 convolutions applied to the 640 12x18 feature maps of the last encoder layer. Because each set of pixels in each feature map at a given $x, y$ coordinate can be thought of as a learned representation of the climate data in a 64x64 patch of the input image, we can think of the 3x3 convolutions as having a local receptive field size of 192x192, so they use a representation of a 192x192 neighborhood from the input image as context to determine the box and object centered in the given 64x64 patch. Our approach is similar to [128] and [219], which use convolutions from small local receptive field filters to regress boxes. This choice is motivated by the fact that extreme weather events occur in relatively small spatiotemporal volumes, with the 'background' context being highly consistent across event types and between events and non-events. This is in contrast to [201], which uses a fully connected layer to consider the whole image as context, appropriate for the task of object identification in natural images, where there is often a strong relationship between background and object.

The bounding box regression loss, $L_{sup}$, is determined as follows:

$$L_{sup} = \frac{1}{N}(L_{box} + L_{conf} + L_{cls}), \tag{4.7}$$

where $N$ is the number of time steps in the minibatch, and $L_{box}$ is defined as:

$$L_{box} = \alpha \sum_i \mathbb{1}_i^{obj} R(u_i - u_i^*) + \beta \sum_i \mathbb{1}_i^{obj} R(v_i - v_i^*), \tag{4.8}$$

where $i \in [0, 216)$ is the index of the anchor box for the i-th grid point, and where $\mathbb{1}_i^{obj} = 1$ if an object is present at the i-th grid point, and 0 if not; $R(z)$ is the smooth L1 loss as used in [206], $u_i = (t_x, t_y)_i$ and $u_i^* = (t_x^*, t_y^*)_i$, $v_i = (t_w, t_h)_i$ and $v_i^* = (t_w^*, t_h^*)_i$, and $t$ is the parametrization defined in [205] such that:

$$t_x = (x - x_a)/w_a, t_y = (y - y_a)/h_a, t_w = \log(w/w_a), t_h = \log(h/h_a)$$

$$t_x^* = (x^* - x_a)/w_a, t_y^* = (y^* - y_a)/h_a, t_w^* = \log(w^*/w_a), t_h^* = \log(h^*/h_a),$$

where $(x_a, y_a, w_a, h_a)$ are the center coordinates and height and width of the closest anchor box, $(x, y, w, h)$ are the predicted coordinates, and $(x^*, y^*, w^*, h^*)$ are the ground truth coordinates.

$L_{conf}$ is the weighted cross-entropy of the log-probability of an object being present in a grid cell:

$$L_{conf} = \sum_i \mathbb{1}_i^{obj}[-\log(p(obj)_i)] + \gamma * \sum_i \mathbb{1}_i^{noobj}[-\log(p(\overline{obj}_i))] \tag{4.9}$$

Finally $L_{cls}$ is the cross-entropy between the one-hot encoded class distribution and the softmax predicted class distribution, evaluated only for predicted boxes at the grid points containing a ground truth box:

$$L_{cls} = \sum_i \mathbb{1}_i^{obj} \sum_{c \in classes} -p^*(c) \log(p(c)) \tag{4.10}$$

The formulation of $L_{sup}$ is similar in spirit to YOLO [201], with a few important differences. Firstly, the object confidence and class probability terms in YOLO are squared-differences between ground truth and prediction, while we use cross-entropy, as used in the region proposal network from Faster R-CNN [206] and the network from [128], for the object probability term and the class probability term respectively. Secondly, we use a different parametrization for the coordinates and the size of the bounding box. In YOLO, the parametrizations for $x$ and $y$ are equivalent to Faster R-CNN's $t_x$ and $t_y$, for an anchor box the same size as the patch it represents (64x64). However $w$ and $h$ in YOLO are equivalent to Faster-RCNN's $t_h$ and $t_w$ for a 64x64 anchor box only if (a) the anchor box had a height and width equal to the size of the whole image and (b) there was no log transform in the faster-RCNN's parametrization. We find both these differences to be important in practice. Without the log term, and using ReLU nonlinearities initialized (as is standard) centered around 0, most outputs (more than half) will give initial boxes that are in 0 height and width. This makes learning very slow, as the network must learn to resize essentially empty boxes. Adding the log term alone in effect makes the "default" box (an output of 0) equal to the height and width of the entire image - this equally slows down learning, because the network must now learn to drastically shrink boxes. Making $h_a$ and $w_a$ equal to 64x64 is a pragmatic 'Goldilocks' value. This makes training much more efficient, as optimization can focus more on picking *which* box contains an object and not as much on what size the box should be. Finally, where YOLO uses squared difference between predicted and ground truth for the coordinate parametrizations, we use smooth L1, due to its lower sensitivity to outlier predictions [206].

### 4.4.6 Experiments and Discussion

#### 4.4.6.1 Framewise Reconstruction

As a simple experiment, we first train a 2D convolutional autoencoder on the data, treating each timestep as an individual training example (everything else about the model is as described in 4.4.5), in order to visually assess reconstructions and ensure reasonable accuracy of detection. Figure 4.17 shows the original and reconstructed feature maps for the 16 climate variables of one image in the training set. Reconstruction loss on the validation set was similar to the training set. As the reconstruction visualizations suggest, the convolutional autoencoder architecture does a good job of encoding spatial information from climate images.

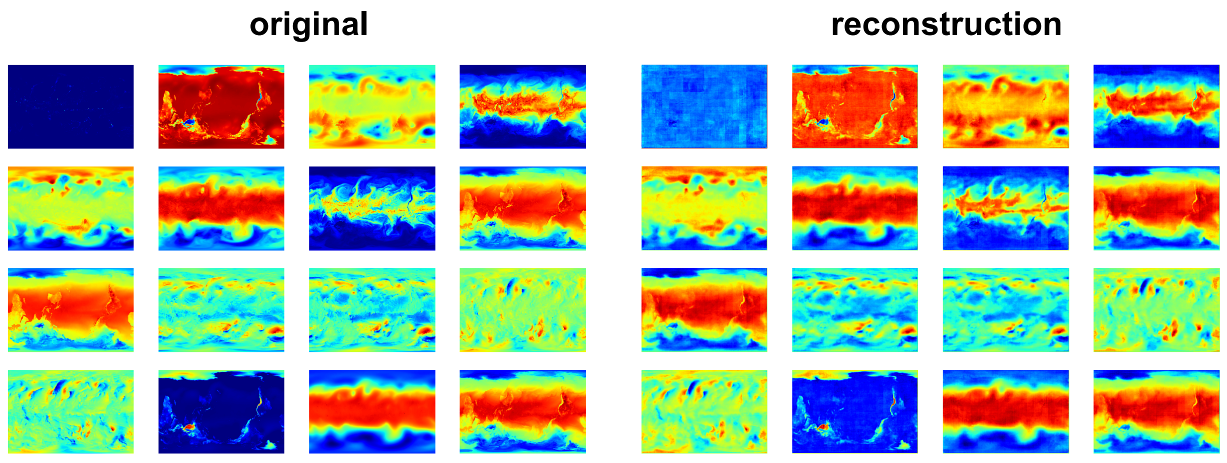


Figure 4.17: Feature maps for the 16 channels in an 'image' from the training set (left) and their reconstructions from the 2D convolutional autoencoder (right).

#### 4.4.6.2 Detection and Localization

All experiments are on ExtremeWeather-small, as described in 4.4.4, where 1979 is train and 1981 is validation. The model is trained with Adam [106], with a learning rate of 0.0001 and weight decay coefficient of 0.0005. For comparison, and to evaluate how useful the time axis is to recognizing extreme weather events, we run experiments with both **2D** (width, height) and **3D** (width, height, time) versions of the architecture described in 4.4.5. Values for $\alpha, \beta, \gamma$ (hyperparameters described in loss Equations 4.8 and 4.9) were selected with experimentation and some inspiration from [201] to be 5, 7 and 0.5, respectively. A lower value for $\gamma$ pushes up the confidence of true positive examples, allowing the model more examples to learn from; this is a way to deal with ground-truth false negatives. Although some of the selection of these parameters is a bit ad-hoc, we assert that our results still provide a good first-pass baseline approach for this dataset. The code is available at `https://github.com/eracah/hur-detect`

During training, we input one day's simulation at a time (8 time steps; 16 variables). The **semi-supervised** experiments reconstruct all 8 time steps, predicting bounding boxes for the 4 labelled timesteps, while the **supervised** experiments reconstruct and predict bounding boxes only for the 4 labelled timesteps. Table 4.11 shows Mean Average Precision (mAP) for each experiment. Average Precision (AP) is calculated for each class in the manner of ImageNet [210], integrating the precision-recall curve, and mAP is averaged over classes. Results are shown for various settings of $\lambda$ (see Equation 4.5) and for two modes of evaluation; at IOU (intersection over union of the bounding-box and ground-truth box) thresholds of 0.1 and 0.5. Because the 3D model has inherently higher capacity (in terms of number of parameters) than the 2D model, we also experiment with higher capacity 2D models by doubling the number of filters in each layer. Figure 4.18 shows bounding box predictions for 2 consecutive (6 hours in between) simulation frames, comparing the 3D supervised vs 3D semi-supervised model predictions.

It is interesting to note that 3D models perform significantly better than their 2D counterparts for ETC and TC (hurricane) classes. This implies that the time evolution of these weather events is an important criteria for discriminating them. In addition, the semi-supervised model significantly improves the ETC and TC performance, which suggests unsupervised shaping of the spatio-temporal representation is important for these events. Similarly, semi-supervised data improves performance of the 3D model (for IOU=0.1), while this effect is not observed for 2D models, suggesting that 3D representations benefit more from unsupervised data. Note that hyperparameters were tuned in the supervised setting, and a more thorough hyperparameter search for $\lambda$ and other parameters may yield better semi-supervised results.

Figure 4.18 shows qualitatively what the quantitative results in Table 4.11 confirm - semi-supervised approaches help with rough localization of weather events, but the model struggles to achieve accurate boxes. As mentioned in 4.4.5, the network has a hard time adjusting the size of the boxes. As such, in this figure we see mostly boxes of size 64x64. For example, for TDs (usually much smaller than 64x64) and for ARs, (always much bigger than 64x64), a 64x64 box roughly centered on the event is sufficient to count as a true positive at IOU=0.1, but not at the more stringent IOU=0.5. This leads to a large dropoff in performance for ARs and TDs, and a sizable dropoff in the (variably-sized) TCs. Longer training time could potentially help address these issues.

| | | | | ETC (46.47%) | TC (39.04%) | TD (9.44%) | AR (5.04%) | |
|---|---|---|---|---|---|---|---|---|
| **M** | **Mode** | **P** | $\lambda$ | **AP (%)** | **AP (%)** | **AP (%)** | **AP (%)** | **mAP** |
| 2D | Sup | 66.53 | 0 | 21.92; 14.42 | 52.26; 9.23 | 95.91; 10.76 | 35.61; 33.51 | 51.42; **16.98** |
| 2D | Semi | 66.53 | 1 | 18.05; 5.00 | 52.37; 5.26 | 97.69; 14.60 | 36.33; 0.00 | 51.11; 6.21 |
| 2D | Semi | 66.53 | 10 | 15.57; 5.87 | 44.22; 2.53 | 98.99; **28.56** | **36.61**; 0.00 | 48.85; 9.24 |
| 2D | Sup | 16.68 | 0 | 13.90; 5.25 | 49.74; **15.33** | 97.58; 7.56 | 35.63; **33.84** | 49.21; 15.49 |
| 2D | Semi | 16.68 | 1 | 15.80; 9.62 | 39.49; 4.84 | **99.50**; 3.26 | 21.26; 13.12 | 44.01; 7.71 |
| 3D | Sup | 50.02 | 0 | 22.65; **15.53** | 50.01; 9.12 | 97.31; 3.81 | 34.05; 17.94 | 51.00; 11.60 |
| 3D | Semi | 50.02 | 1 | **24.74**; 14.46 | **56.40**; 9.00 | 96.57; 5.80 | 33.95; 0.00 | **52.92**; 7.31 |

Table 4.11: 2D and 3D supervised and semi-supervised results, showing Mean Average Precision (mAP) and Average Precision (AP) for each class, at IOU=0.1 and IOU=0.5. **M** is a model and **P** is millions of parameters; and $\lambda$ weights the amount that reconstruction contributes to the overall loss.

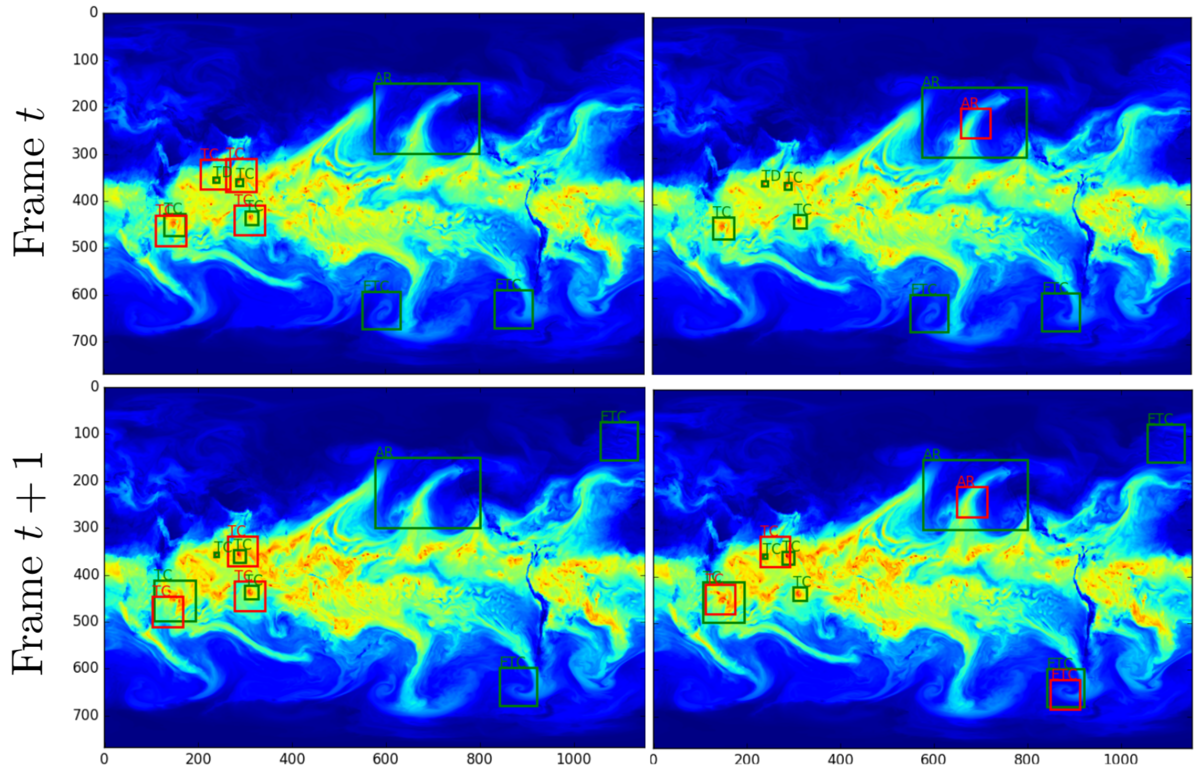


Figure 4.18: Bounding box predictions shown on 2 consecutive (6 hours in between) simulation frames, for the integrated water vapor column channel. Green = ground truth, Red = high confidence predictions (confidence above 0.8). 3D supervised model (Left) and semi-supervised (Right).

### 4.4.6.3 Feature exploration

In order to explore learned representations, we use t-SNE [134] to visualize the autoencoder bottleneck (last encoder layer). Figure 4.19 shows the projected feature maps for the first 7 days in the training set for both 3D supervised (top) and semi-supervised (bottom) experiments. Comparing the two, it appears that more TCs (hurricanes) are clustered by the semi-supervised model, which would fit with the result that semi-supervised information is particularly valuable for this class. Viewing the feature maps, we can see that both models have learned spiral patterns for TCs and ETCs.

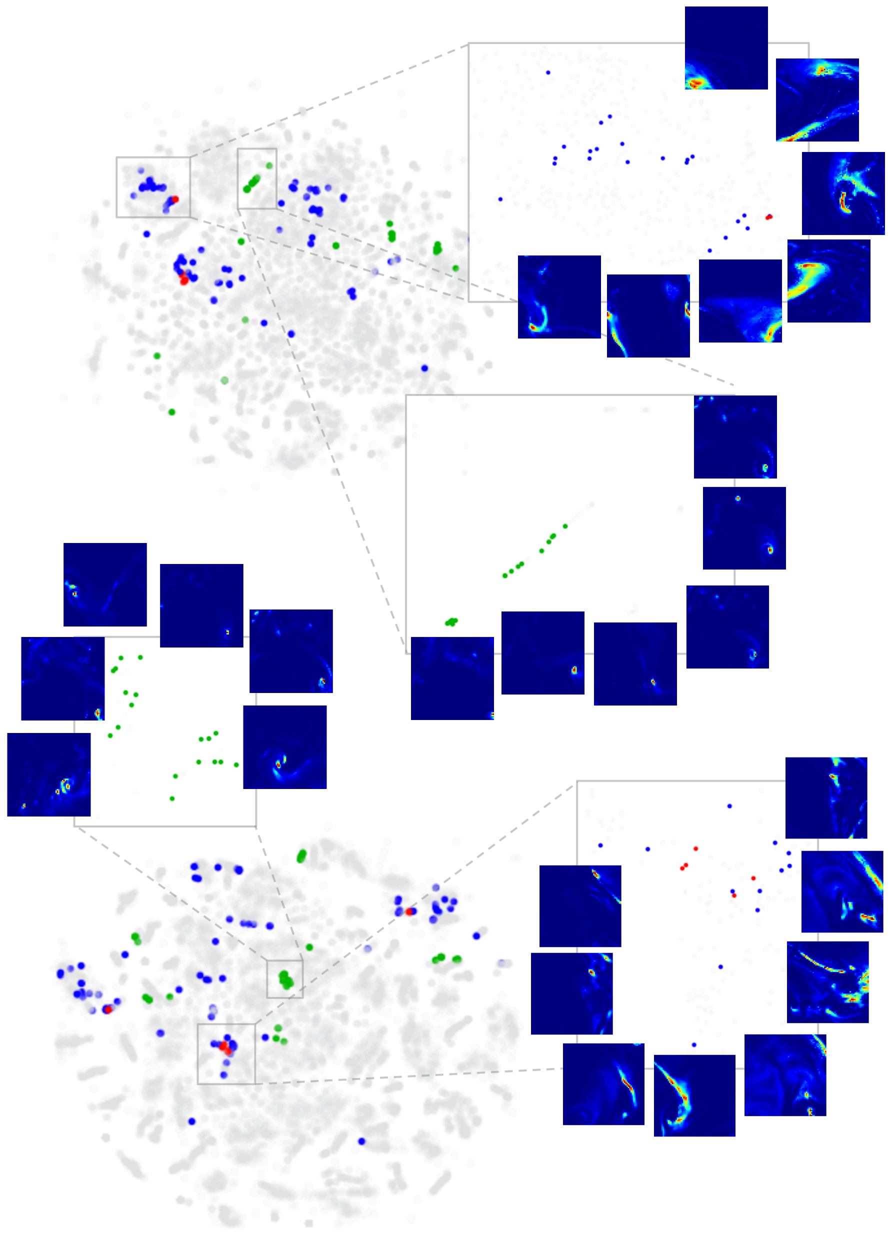


Figure 4.19: t-SNE visualisation of the first 7 days in the training set for 3D supervised (top) and semi-supervised (bottom) experiments. Each frame (time step) in the 7 days has 12x18 = 216 vectors of length 640 (number of feature maps in the code layer), where each pixel in the 12x18 patch corresponds to a 64x64 patch in the original frame. These vectors are projected by t-SNE to two dimensions. For both supervised and semi-supervised, we have zoomed into two dense clusters and sampled 64x64 patches to show what that feature map has learned. Grey = unlabelled, Yellow = tropical depression (not shown), Green = TC (hurricane), Blue = ETC, Red = AR.

### 4.4.7 Conclusions and Future Work

We introduce to the community the ExtremeWeather dataset in hopes of encouraging new research into unique, difficult, and socially and scientifically important datasets. We also present a baseline method for comparison on this new dataset. The baseline explores semi-supervised methods for object detection and bounding box prediction using 3D autoencoding CNNs. These architectures and approaches are motivated by finding extreme weather patterns, a meaningful and important problem for society. Thus far, the climate science community has used hand-engineered criteria to characterize patterns. Our results indicate that there is much promise in considering deep learning based approaches. Future work will investigate ways to improve bounding-box accuracy, although even rough localizations can be very useful as a data exploration tool, or initial step in a larger decision-making system. Further interpretation and visualization of learned features could lead to better heuristics and understanding of the way different variables contribute to extreme weather events. Insights in this paper come from only a fraction of the available data, and we have not explored such challenging topics as anomaly detection, partial annotation detection, and transfer learning (e.g. to satellite imagery). Moreover, learning to generate future frames using GAN's [59] or other deep generative models, while using performance on a detection model to measure the quality of the generated frames, could be another very interesting future direction. We make the ExtremeWeather dataset available in hopes of enabling and encouraging the machine learning community to pursue these directions. The retirement of Imagenet [211] marks the end of an era in deep learning and computer vision. We believe the era to come should be defined by data of social importance, pushing the boundaries of what we know how to model.

## 4.5 Deep Learning for Segmentation

### 4.5.1 Collaborators

The research in this section was originally published in [115]. The collaborators were **Thorsten Kurth**, **Sean Treichler**, **Everett Phillips**, **Massimiliano Fatica**, **Joshua Romero**, **Ankur Mahesh**, **Mayur Mudigonda**, **Michael Matheson**, **Jack Deslippe**, and **Michael Houston**.

### 4.5.2 Pattern Detection for Characterizing Extreme Weather

In this work, we use the TensorFlow [138, 239] deep learning framework, which allows the programmatic definition of even very complicated network *graphs* in tens of lines of Python code. TensorFlow provides portability with its capability to map a graph onto multi- and many-core CPUs as well as GPUs. Due to the heavy use of linear algebra-based primitives (e.g. convolutions), most networks (including ours) perform very well on GPUs. The graph also captures the parallelism available in the computation, and TensorFlow uses a dynamic scheduler to select which operation (or *layer*) to compute based on the availability of inputs. (Scheduling is performed independently on each process in a distributed job, leading to challenges with collective communication as described in 5.3.4.1.3.)

A deep learning model is trained by comparing its output to known *labels*, using a *loss function* to quantify the differences between the two. The model parameters (e.g. convolution weights) form a very-high-dimensional (typically millions) space, and training becomes an optimization problem to find the point in the space that minimizes the loss. Each layer is differentiable by construction, and variants of gradient descent are typically used for optimization. Since training sets can be very large, the most common variant is stochastic gradient descent, which performs updates based on randomly selected subsets (or *batches*) of the overall training set.

To parallelize training, a common technique is to replicate the model across ranks, with each rank processing a different local batch of images. Updates to the model are aggregated between ranks during each training step. As a deep learning problem is "scaled out," the size of the global batch (combined batch across all ranks) grows, but the size of the overall training set remains fixed, making it distinct from traditional weak scaling. The effect of batch size on convergence rate is not fully understood, but with the right *hyperparameters* (parameters for the optimizer rather than the model), larger batches require fewer steps to converge, improving overall time to solution if the parallel efficiency is sufficiently high. Although an analogous form of strong scaling also exists (keeping the global batch size constant as worker count grows), it is generally only of interest when effective hyperparameters cannot be found for a larger global batch size.

#### 4.5.2.1 Segmentation Architecture

High-level frameworks like TensorFlow make it convenient to experiment with different networks. We evaluated two very different networks for our segmentation needs. The first is a modification of the *Tiramisu* network [93]. Tiramisu is an extension to the Residual Network (ResNet) architecture [74] which introduced *skip connections* between layers to force the network to learn residual corrections to layer inputs. Where ResNet uses addition, Tiramisu uses concatenation to combine the inputs of one or more layers of the network with their respective outputs. The Tiramisu network is comprised of a *down path* that creates an *information bottle-neck* and an *up path* that reconstructs the input. To perform pixel-level segmentation, Tiramisu includes skip connections spanning the down and up paths to allow re-introduction of information lost in the down path. Our Tiramisu network uses five dense blocks in each direction, with 2,2,2,4 and 5 layers respectively (top to bottom). We then train the model using adaptive moment estimation (ADAM) [106].

The second network we evaluated is based on the recent DeepLabv3+ network [21] and is shown in Figure 4.20. DeepLabv3+ is an *encoder-decoder architure* that uses well-proven networks (in our case ResNet-50) as a core. The *encoder* performs a function similar to Tiramisu's down path but avoids loss of information by replacing some of the downscaling with *atrous convolution.* Atrous convolutions sample the input sparsely according to a spec-

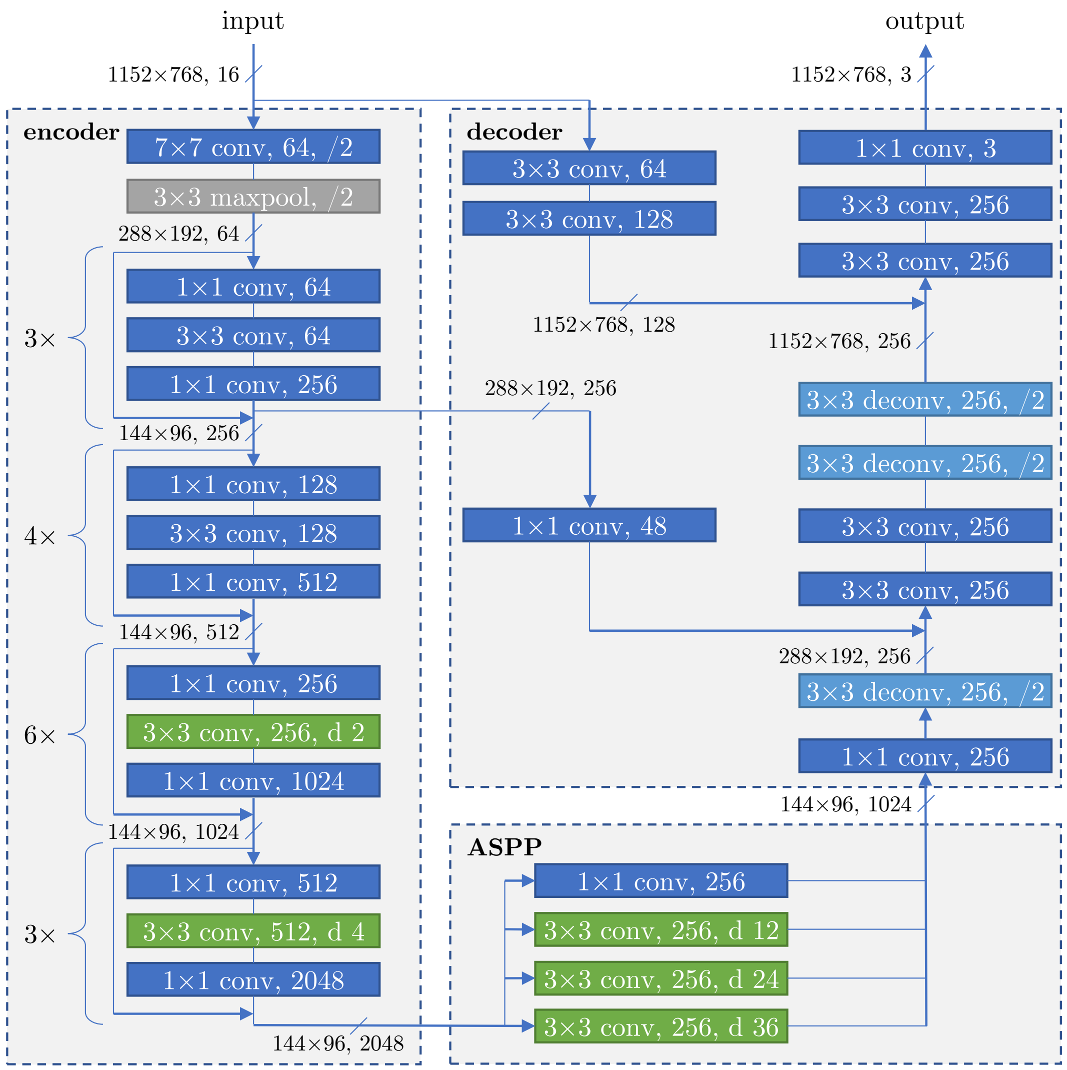


Figure 4.20: Schematic of the modified DeepLabv3+ network used in this work. The encoder (which uses a ResNet-50 core) and atrous spatial pyramid pooling (ASPP) blocks are changed for the larger input resolution. The DeepLabv3+ decoder has been replaced with one that operates at full resolution to produce precise segmentation boundaries. Standard convolutions are in dark blue, and deconvolutional layers are light blue. Atrous convolution layers are in green and specify the *dilation* parameter used.

ified *dilation* factor to detect larger features. This simplifies the *decoder* (corresponding to Tiramisu's up path) considerably. Our modifications to these existing networks are described in 4.5.3.4.

#### 4.5.2.2 Climate Dataset and Ground Truth Labels

We utilize 0.25-degree Community Atmosphere Model (CAM5) output for this study. Climate variables are stored on an 1152×768 spatial grid, with a temporal resolution of 3 hours. Over 100 years of simulation output are available in HDF5 files. Ideally, climate scientists would hand-label pixel masks corresponding to events. In practice, scientists currently use a combination of heuristics to produce masks on large datasets. The first step is to process climate model output with the Toolkit for Extreme Climate Analysis [187, 184] to identify TCs. A floodfill algorithm is used to create spatial masks of ARs [223], which provides the labels for our training process.

There are about 63K high-resolution samples in total, which are split into 80% training, 10% test and 10% validation sets. We use all available 16 variables (water vapor, wind, precipitation, temperature, pressure, etc). The pixel mask labels correspond to 3 classes: Tropical Cyclone (TC), Atmospheric River (AR) and background (BG) class.

### 4.5.3 Deep Learning Innovations

**4.5.3.0.1 Weighted loss** The image segmentation task for climate analysis is challenging because of the high class imbalance: about 98.2% of the pixels are BG and about 1.7% of the overall pixels are ARs. Pixels labelled as TCs make up less than 0.1% of the total. With an unweighted loss function, each pixel contributes equally to the loss function, and a network can (and did, in practice) achieve high accuracy (98.2% in our case) by simply predicting the dominant background class for all pixels. To improve upon this situation, we use a weighted loss calculation in which the loss for each pixel is weighted based on its labeled class. The per-pixel weight map is calculated as part of the input processing pipeline and provided to the GPU along with the input image. Our initial experiments used the inverse of the class frequencies for weights, attempting to equalize the collective loss contribution from each class. We found that this approach led to numerical stability issues, especially with FP16 training, due to the large difference in per-pixel loss magnitudes. We examined more moderate weightings of the classes and found that using the inverse square root of the frequencies addressed stability concerns while still encouraging the network to learn to recognize the minority classes (see Figure 4.21).

#### 4.5.3.1 LARC

Layer-wise adaptive rate control (LARC) [54] is designed to control the magnitude of weight updates by keeping them small compared to the norm of layer's weights. LARC uses a separate independent learning rate for every layer instead of every weight. The magnitude of the update is defined with respect to the weight's norm. LARC improves the accuracy of large networks, especially when trained using large batch sizes. Compared to layer-wise adaptive rate scaling (LARS) [276], LARC removes the need for complex learning rate warm-up techniques and is thus much easier to use. Given all these advantages, we use LARC for the results reported in this study.

#### 4.5.3.2 Multi-channel segmentation

Traditional image segmentation tasks work on 3-channel RGB images. However, scientific datasets can be comprised of many channels: in case of the CAM5 climate dataset, those can incorporate fields such as temperature, wind speeds, pressure values, and humidity at different altitudes. Our initial experiments on Piz Daint used 4 channels that were thought to be the most important, but when the network was moved to Summit, the additional computational capabilities allowed the use of all 16 channels, which improved the accuracy of the models dramatically. The optimal subset of channels to use likely lies in between these two, and we plan to take advantage of the ability to rapidly train this network at scale to tune for the right subset.

#### 4.5.3.3 Gradient lag

Most of the all-reduce operations required for gradient computation can be overlapped with other computation, but the top-most layer's gradient computation is a sequential bottleneck for a standard optimizer. The network-induced latency of this computation can limit performance at large scale. To improve parallel efficiency, we modified the optimizer to use the gradients computed in the previous step when performing weight updates. In addition to improving the overlap of communication and computation, this *lagging* of the gradients allows Horovod to more efficiently batch the tensors for all-reduce computations, increasing network throughput. Although a change to the optimizer usually requires changes to the hyperparameters to maintain convergence properties, the performance benefit is usually worth the effort at large scale. A similar gradient lagging strategy, known as elastic averaging SGD (EASGD), was shown to be effective, with even larger degrees of lag [279].

#### 4.5.3.4 Modifications to the neural network architectures

The developers of the original Tiramisu network advocate the use of many layers with a relatively small growth rate per layer (e.g. 12 or 16) [93] and our initial network design used a growth rate of 16. This network learned well, but performance analysis of the resulting TensorFlow operations on Pascal and Volta GPUs found considerable room for improvement, and we determined that a growth rate of 32 would be significantly more efficient. To keep the overall network size roughly the same, we reduced the number of layers in each dense block by a factor of two and changed the convolutions from $3 \times 3$ to $5 \times 5$ to maintain the same receptive field. Not only was the new network much faster to compute, we found that it trained faster and yielded a better model than our original network.

For DeepLabv3+, the atrous convolutions result in a more computationally expensive network than Tiramisu. The standard DeepLabv3+ design makes the compromise of performing segmentation at one-quarter resolution (i.e. $288 \times 192$ rather than $1152 \times 768$) to keep the computation tractable for less-powerful systems, at the cost of fidelity in the resulting masks. The irregular and fine-scale nature of our segmentation labels requires operating at the native resolution of the dataset. With the unparalleled performance of Summit available for this work, we were able to replace the standard DeepLabv3+ decoder with one that operates at full resolution, thereby benefiting the science use case.

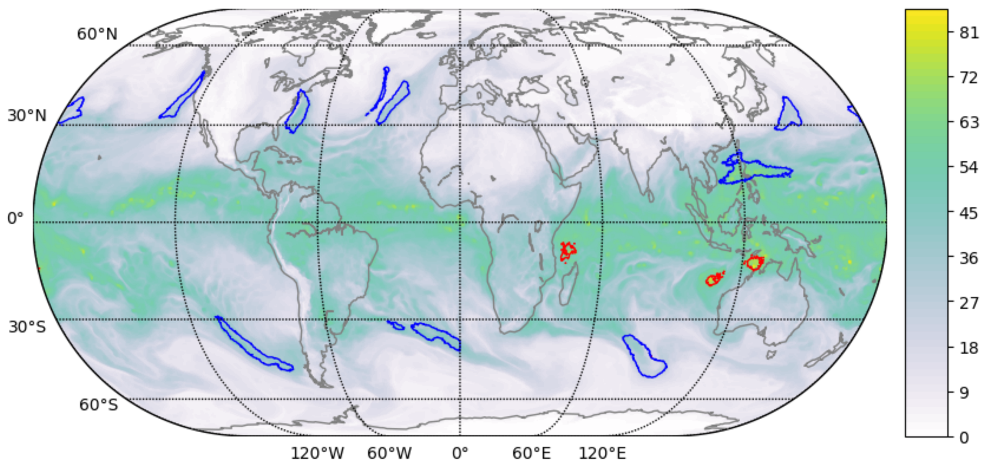
### 4.5.4 Segmentation Results

Segmentation accuracy is often measured using the *intersection over union* (IoU) metric. The Tiramisu network obtained an IoU of 59% on our validation data set, while our modified DeepLabv3+ network was able to achieve 73% IoU. Visually, this translates into qualitatively pleasing masks as seen in Figure 4.21. Not only does the network find the same atmospheric features, it also makes a very good approximation of their exact boundaries. In some cases, the boundaries predicted by the model appear to be superior to the labels provided by heuristics. One of the tropical cyclones in Figure 4.21b does suffer from overprediction. This is an expected consequence of our weighted loss function, which penalizes a false negative on a TC by roughly $37\times$ more than a false positive.

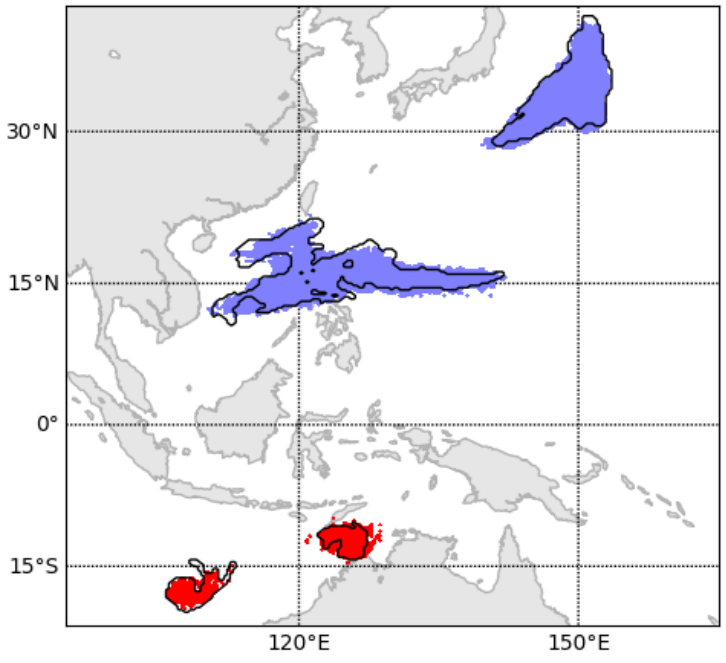
### 4.5.5 Conclusions

This is the first successful demonstration of the application of Deep Learning for extracting pixel-level segmentation masks in the climate science community. This analysis opens the door for more sophisticated characterization of extreme weather than what has been possible before. Prior to this work, climate scientists reported coarse summary statistics such as number of global storms. In contrast, we can now compute conditional precipitation, wind velocity profiles and power dissipation indices for *individual* storm systems. These sophisticated metrics will enable us to characterize the *impact* from each event (physical damage to infrastructure, flooding, monetary loss, etc) with unprecedented fidelity.

In the future, we will explore advanced architectures that can consider temporal evolution of storms. This will increase the resident size of the network architecture, requiring model (as well as data) parallelism for efficient execution. We also plan on working with the climate science community to generate high quality ground truth datasets without resorting to heuristics. Developing accessible interfaces for specifying masks, and bootstrapping the process using online, semi-supervised methods, is an area for further investigation.



(a) Segmentation masks overlaid on a globe. Colors (white->yellow) indicate IWV (integrated water vapor, $kg/m^2$), one of the 16 input channels used by the network.



(b) Detailed inset showing predictions (red and blue) vs. labels used in training (black).

Figure 4.21: Segmentation results from modified DeepLabv3+ network. Atmospheric rivers (ARs) are labeled in blue, while tropical cyclones (TCs) are labeled in red.

# Chapter 5

# Scaling Methods to Large Climate Datasets

Climate simulations provide us with an unprecedented view of the state of earth's present, and potential future climate under global warming. State of the art climate codes, such as the Community Atmosphere Model (CAM v5) [28], when run in 25-km spatial resolution with 6-hour data dumps, produce over 100TB from a single 25-year integration period. The current CMIP-5 archive [31], consisting of international contributions from a number of climate modeling groups consists of over 5PB of data; this dataset was mined extensively for the IPCC AR5 report [89]. It is anticipated that CMIP-6 dataset [259] will cross the exabyte threshold with 25-km model runs being the norm. Faced with this massive deluge of multi-variate, spatio-temporal data, sophisticated and scalable "pattern detection" tools are critical for extracting meaningful scientific insights. In this Chapter, we demonstrate state-of-the-art progress in scaling methods based on heuristics, and Deep Learning to massive datasets, on modern HPC platforms.

## 5.1 Scaling TECA

### 5.1.1 Collaborators

The research in this section was originally published in [184]. The collaborators were **Surendra Byna**, **Venkatram Vishwanath**, **Eli Dart**, **Michael Wehner**, and **William D. Collins**.

### 5.1.2 Introduction

Thus far, we have applied the TECA [187] framework to identify three different classes of storms: tropical cyclones, atmospheric rivers and extra-tropical cyclones. Due to the high-frequency nature of the data required to identify and track individual storms in a climate model simulation, the raw input datasets that we have analyzed range from 0.5TB to 13TB. As the entire climate modeling community starts to upgrade their infrastructure to run at comparably high resolutions (25 km or better), we expect the publicly available datasets necessary for this type of analysis to exceed 10PB. Manual labeling and extraction of patterns at such scales is simply impossible, thereby requiring the development and application of "Big Data" analytics methods. The techniques that we have developed are amenable to parallel execution, and are demonstrated to scale up to full size of the largest machines available to us, including a 150,00 core Cray XE6 and 750,000 core IBM BG/Q platform.

### 5.1.3 TECA : Toolkit for Extreme Climate Analytics

TECA [187] is a climate-specific, high-performance pattern detection toolkit that is designed for efficient execution on HPC systems. We have developed the code in C/C++, and utilize MPI for inter-process communication. We utilize NetCDF-4 for parallel reads, and MPI-IO for storing results.

The design and implementation of TECA is based on our first hand experience with pattern detection problems in climate science. After analyzing the climate pattern detection literature for a number of event types, we discovered the following "design pattern". The detection process can be typically broken down into two steps:

1. Detection of candidate points that satisfy multi-variate constraints
2. Stitching of candidate points into a trajectory that satisfies spatio-temporal constraints

Step 1 tends to be data-intensive, involving loading anywhere between 10GB-10TB of data. The algorithm has to scan through all of the relevant fields to select candidate points. However, this step can be executed in parallel across timesteps. The degree of parallelism can be as high as the number of timesteps in the processed dataset (typically $10^2$-$10^5$); hence a dramatic speedup in overall runtime is feasible. Step 2 involves pairwise analysis on potential storm matches across consecutive time slices in order to stitch trajectories. However, a small amount of data (typically 10MB-1GB) is required for this analysis, and this can be easily loaded on memory on a single node and executed in serial.

Conceptually, Steps 1 and 2 can be directly translated to the MapReduce computational paradigm powering much of the commercial Big Data Analytics workloads. TECA implements a custom framework for processing scientific datasets, with an eye towards high performance. We utilize the Message Passing Interface (MPI) for optimizing job launch; communication and synchronization traffic. In the instance of multi-model archives, sub-communicators corresponding to individual models and ensemble members are created and initialized, thereby minimizing synchronization with other tasks. NetCDF files are striped across multiple low-level storage targets to optimize read performance. Writing (relatively small) partial results in Step 2 can create metadata bottlenecks, especially at concurrencies in excess of 50,000 cores. We implement a 2-phase collective I/O mechanism to aggregate writes on a smaller ($O(1000)$) number of nodes and perform file-per-node writes using MPI-IO. The cumulative effect of these best practices in HPC and Parallel I/O, is that we are able to successfully run TECA jobs at full concurrency on petascale platforms.

All of these detection and stitching criteria can be easily accommodated within the design of TECA; thereby utilizing parallel job launch, execution and parallel I/O capabilities. Apart from returning summary statistics on storm counts and location, we are also able to pull out valuable detailed information on precipitation patterns and velocity profiles of storms during the course of their lifetime.

#### 5.1.3.1 TECA for detecting Tropical Cyclones

We have implemented the Tropical Cyclone (TC) detection procedure outlined in [109]. The detection step consists of finding co-located vorticity maxima, pressure minima (within a radius of 5°) and temperature warm-core centers. Splines are fitted to relevant fields, which are defined on latitude-longitude grids. Local maxima and minima of functions are computed by performing a line search on the splines within the prescribed spatial window. Once potential storm center candidates have been identified, the next step is to impose spatial and temporal constraints to connect the storm centers into a trajectory. In the case of TCs, the stitching step involves linking storms across subsequent 6-hr time windows. Candidate storms are prescribed to travel less than 400km in 6-hrs, persist for at least 2 days, and have a wind velocity (greater than 17m/s) during at least 2 days within their lifetime.

#### 5.1.3.2 TECA for detecting Extra-Tropical Cyclones

We implement the Extra-Tropical Cyclone (ETC) detection and tracking procedure documented in [256]. We detect a local minima in the pressure field within a 100x100 km radius. Ties between adjacent low-pressure storm centers are resolved based on strength of the local laplacian (i.e. storm centers with largest laplacian are declared to be storm center candidates). Potential candidates are stitched into trajectories by performing a nearest neighbor analysis with distance constraints: storms are restricted to travel less than 1000km in a 6-hr window, and less than 700 km in a 6-hr window in the North, South, and Westward directions. We only retain storms that persist for more than 24 hours, and travel greater 500km over their lifetime. Storms over high elevation areas (greater than 1500km) are excluded.

#### 5.1.3.3 TECA for detecting Atmospheric Rivers

Atmospheric Rivers (AR) are large, spatially coherent weather systems with high concentrations of elevated water vapor. These systems often cause severe downpours and flooding over the western coastal United States and western Europe. We have implemented an algorithm to detect ARs in the TECA framework [13]. We first compute a 2D Integrated Water Vapor (IWV) by performing a vertical integral on the specific humidity field. Following the definition of physical features of an AR [194]; we perform a thresholding operation for identifying all grid points with IWV greater than 2cm. We then use a connected component labeling algorithm to find all the connected regions of grid points. We test if a candidate originates in the tropics, and makes landfall on the US coast. For all the polygons satisfying the origin and the landfall conditions, we compute a medial axis, and check if the length of the AR greater than 2000km and if the width of the AR less than 1000km. If a polygon satisfies all of these geometric constraints, we declare it to be an atmospheric river.

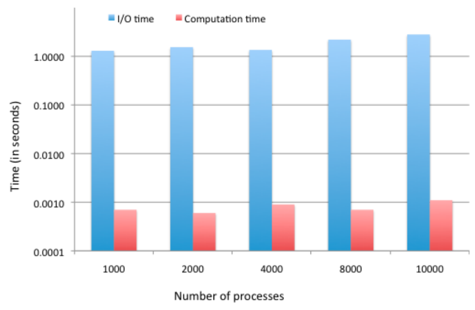


Figure 5.1: Weak Scaling plots for TECA AR analysis.

**5.1.3.3.1 Weak Scaling** Figure 5.1 shows results from our weak scaling experiment. The x-axis shows number of MPI processes, and the y-axis shows the time in seconds (in logarithmic scale). To recall the experimental setup, each process analyzes data for a single day; as more processes as added, the detection algorithm works on a proportionally larger number of days.

We observe that the majority of the execution time of our tool is dominated by I/O ( 98%). Since each process only works on one day's worth of data, we expect the the I/O time and the computation time to remain constant as the number of processes increases. While these costs stayed relatively constant, we noticed a small increase in the observed time. We attribute this to a small fraction of MPI processes taking longer than others to finish their processing. As the number of processes increase from 1000 to 10000 processes, the observed computation time fluctuates between 0.7ms and 1.1ms, we believe that these fluctuations are random in nature (and not systematic).

The I/O time also increases slightly; the main reason for this increase is due to shared access to the same input file for reading filenames. As the number of processes increase, the time to read the file names increase from 0.29s for 1000 processes to 1.54s for 10,000 processes. In the same tests, the time to read the integrated water vapor data remains about the same, 1.01s 1000 processes and 1.32s for 10,000 processes.

In Figure 5.2, we show the aggregate I/O throughput against the number of processes. While the previous figure (Figure 5.1) shows I/O time for the process that took the longest time, this figure shows aggregate I/O bandwidth of all processes. We calculated the aggregate I/O throughput as the sum of I/O throughput at each process. Since each process runs independently without any synchronization, measuring global I/O bandwidth for the appli-

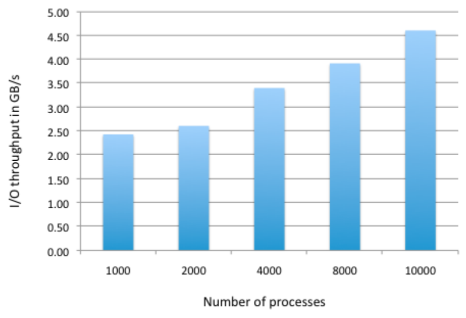


Figure 5.2: I/O performance associated with TECA AR weak scaling runs.

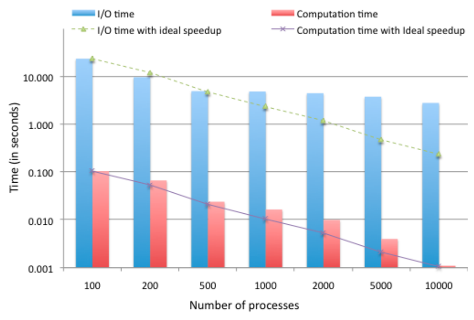


Figure 5.3: Strong Scaling plots for TECA AR analysis.

cation does not reflect I/O performance of the tool. As the number of processes increases, the I/O throughput also increases. In the model dataset each file contains five days of data and is therefore shared by five processes. Since each read request fetches a full stripe of data (1 MB in Lustre file system configured on Hopper), this 1 MB stripe of data includes all water vapor data for all five days. This explains achieving good I/O performance even though each process is expected to read only about 128 KB of data.

Another reason for the increased I/O throughput is because as more processes are making I/O requests to the file system, they collectively occupy more attention of the file system. Therefore, the observed throughput is a larger fraction of the file system's peak performance. In our tests, we see the aggregate I/O throughput increase from 2.5 GB/s with 1000 processes to about 4.6 GB/s with 10,000 processes.

**5.1.3.3.2 Strong Scaling** Figure 5.3 shows strong scaling results with a fixed data size related to 10,000 days. This experiment shows how the application scales as the number of processes increase, while the data size is fixed. The two bars show the I/O time and the computation time. The sum of these two costs is equal to the total execution time of the algorithm. The upper trend line (dashed) refers to the I/O time if ideal speedup were achieved and the lower trend line represents computation time if ideal speedup were achieved. We calculated the time with ideal speedup in reference to the measured time when 100 processes were used. For instance, if the I/O overhead with 100 processes is $t$, then the I/O overhead with 200 processes is $t/2$ and that with 500 processes is $t/5$, and so on. The combination of reading file names from input file and the reading vapor data from climate data set dominate the overall execution time. The total I/O time constitutes 99% of the execution time.

In Figure 5.3, we see that the computation time speedup generally agrees with ideal scaling. This suggests that the computations are relatively load-balanced and amenable to parallelization. In this case, each process handles data from a number of different days, which minimizes the effect of random fluctuations discussed earlier.

The I/O times are very close to ideal speedup for the test cases with 100, 200 and 500 processes. As indicated before, five processes read from a single data file and their read operations are most likely served by a single disk read, which means that 100 OSTs can serve 500 processes. In going from 100 to 500 processes, our program is effectively using more OSTs from the file system, therefore the I/O time scales well. As more processes are used, it is no longer possible to have each OST serve five processes. This creates I/O contention and increases the time needed to complete the I/O operations. We see that the I/O time in Figure 5.3 goes above the expected value for ideal speedup when more than 1000 processes are used.

In Figure 5.4, we show the aggregate I/O throughput as the number of processes increases with fixed data size. The I/O throughput increases as the number of processes increase up to 500 processes following the linear trend. From the 1000 processes case, the I/O throughput falls short of ideal growth. Nevertheless, the aggregate throughput still increases, reaching 4.6 GB/s with 10,000 processes.

Our results indicate that our tool is quite well suited for weak scaling. Our tool can analyze more data with a larger number of processes. This will be useful in processing output data from century scale climate model integrations.

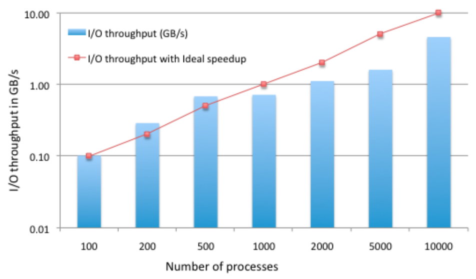


Figure 5.4: I/O performance associated with TECA AR strong scaling runs.

### 5.1.4 Experimental Setup

#### 5.1.4.1 Data

We utilized multi-model output from the community produced CMIP-5 archive [31], and a high-resolution version of the Community Atmospheric Model (CAM5) [28] simulations conducted by our group at NERSC. The CMIP-5 datasets are freely and publicly accessible via a number of international Earth System Grid Federation web portals and are the basis of most climate model results presented in the IPCC AR5 WG1 report [44]. The observational SSM/I datasets are available via a web portal [204]. In all cases, the datasets are available as multiple netcdf files. Each file typically contains all relevant 2D variables and spans one year's worth of data.

#### 5.1.4.2 Platforms

We utilized the Hopper system at NERSC, and the Mira system at ALCF for all results reported in this paper. Hopper is a 1.28 PF, Cray XE6 system featuring 153,216 compute cores, 212TB of memory and 2PB of disk available via a 35 GB/s Lustre filesystem. Mira is a 10PF, IBM BG/Q system featuring 786,432 cores, 768 TB of memory with 384 I/O nodes accessible via GPFS.

### 5.1.5 Results

We now report on both the scaling performance obtained by TECA on various HPC platforms, as well as the scientific results facilitated by these runs.

#### 5.1.5.1 Scaling Performance

Table 1 summarizes the performance of TECA on a range of pattern detection problems. We analyzed CAM5 model output (0.5-13 TB), CMIP-5 multi-model output (6 TB), and the SSMI (35 GB) satellite data product. We ran TECA at full scale on Hopper and Mira platforms, facilitating pattern detection on these massive datasets. Needless to say, such pattern detection problems cannot be tackled on individual workstations in a reasonable amount of time.

### 5.1.6 Limitations and Future Work

This work is one of the leading example of the use of high performance computing for solving pattern "detection" problems. The definition of a pattern is assumed to be conveniently prescribed by various experts in the climate science community. While this is true for the three events that we examined, a much broader class of events exist (e.g. weather fronts, mesoscale convective systems, derechos, blocking events), for which there does not exist an algorithmic definition that could be implemented in TECA. Such problems are more

| Climate Pattern | Dataset | Dataset Size | Serial runtime (Estimated) | Parallel runtime | Concurrency | Platform |
|---|---|---|---|---|---|---|
| TC | CAM5 1° | 0.5 TB | $\approx$ 8 years | 31 min | 149,680 cores | Hopper |
| TC | CAM5 0.25° | 13 TB | $\approx$ 9 years | 58 min | 80,000 cores | Hopper |
| AR | SSM/I | 35 GB | $\approx$ 11 hours | 5 sec | 10,000 cores | Hopper |
| ETC | CMIP-5 | 6 TB | $\approx$ 10 years | 95 min | 755,200 cores | Mira |

Table 5.1: TECA is able to scale effectively on various HPC platforms, dataset sizes and extreme weather patterns. The table highlights some of the largest scale runs attempted.

amenable to the classic machine learning paradigm, wherein a modest number of training examples would be provided by human experts, and a learning algorithm could determine the relevant features and inter-relationships that "define" the extreme weather pattern. We are currently examining the use of Deep Learning methods, as a complementary approach to TECA for targeting a much broader class of weather patterns.

### 5.1.7 Conclusions

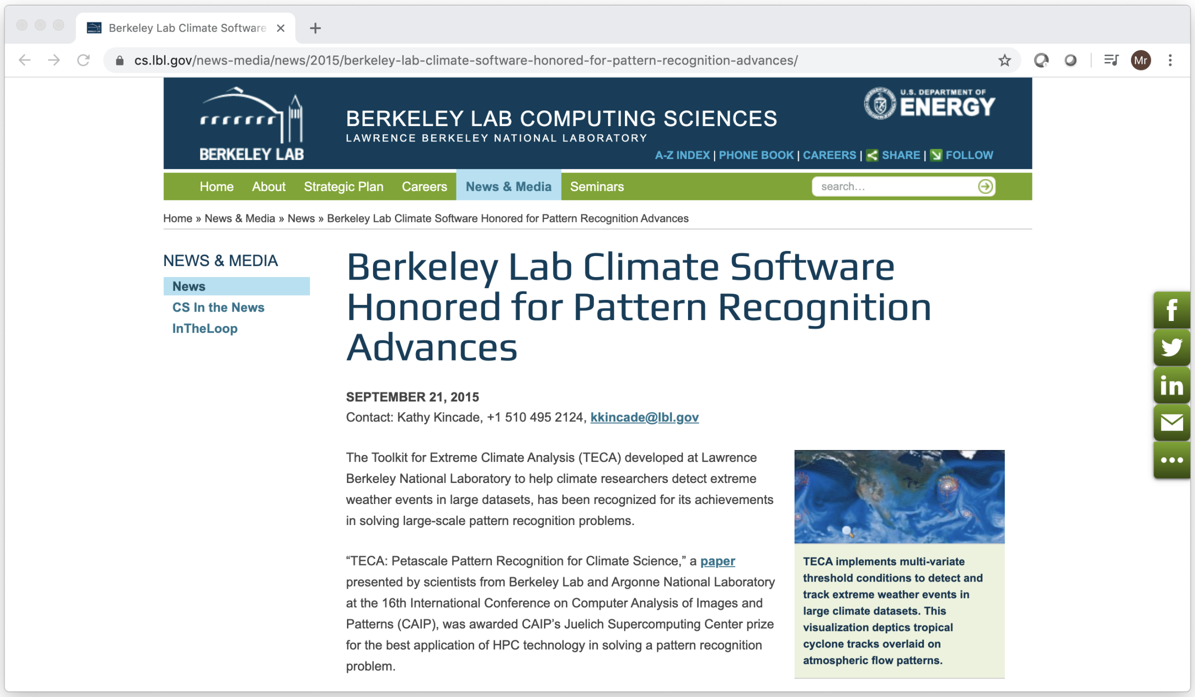
Pattern recognition problems are increasingly common in the scientific world. As a leading example, climate science requires sophisticated pattern recognition on TB-PB sized datasets. We have developed and successfully applied TECA to the problem of finding extreme weather phenomena (such as tropical cyclones, atmospheric rivers and extra-tropical cyclones) across most contemporary climate models (CAM5), data archives (CMIP-5) and observational products (SSMI). We have scaled TECA on DOE's leading HPC platforms at NERSC and ALCF, and obtained important scientific insights on the potential change in extreme weather phenomena in future climate regimes.

TECA is the most scalable software for implementing and applying heuristics to massive climate datasets. This feat was acknowledged by the Juelich Supercomputing Center prize at a pattern recognition conference in 2015.

## 5.2 Scaling Deep Learning on CPUs

### 5.2.1 Collaborators

The research in this section was originally published in [114]. The collaborators were Thorsten Kurth, Jian Zhang, Nadathur Satish, Ioannis Mitliagkas, Evan Racah, Mostofa Patwary, Tareq Malas, Narayanan Sundaram, Wahid Bhimji, Jack Deslippe, and Pradeep Dubey.


Berkeley Lab Climate Software
cs.lbl.gov/news-media/news/2015/berkeley-lab-climate-software-honored-for-pattern-recognition-advances/
BERKELEY LAB
BERKELEY LAB COMPUTING SCIENCES
LAWRENCE BERKELEY NATIONAL LABORATORY
U.S. DEPARTMENT OF ENERGY
A-Z INDEX | PHONE BOOK | CAREERS | SHARE | FOLLOW
Home About Strategic Plan Careers News & Media Seminars
search...
Home » News & Media » News » Berkeley Lab Climate Software Honored for Pattern Recognition Advances
NEWS & MEDIA
News
CS In the News
InTheLoop

Berkeley Lab Climate Software Honored for Pattern Recognition Advances

SEPTEMBER 21, 2015
Contact: Kathy Kincade, +1 510 495 2124, kkincade@lbl.gov

The Toolkit for Extreme Climate Analysis (TECA) developed at Lawrence Berkeley National Laboratory to help climate researchers detect extreme weather events in large datasets, has been recognized for its achievements in solving large-scale pattern recognition problems.

"TECA: Petascale Pattern Recognition for Climate Science," a paper presented by scientists from Berkeley Lab and Argonne National Laboratory at the 16th International Conference on Computer Analysis of Images and Patterns (CAIP), was awarded CAIP's Juelich Supercomputing Center prize for the best application of HPC technology in solving a pattern recognition problem.

TECA implements multi-variate threshold conditions to detect and track extreme weather events in large climate datasets. This visualization deptics tropical cyclone tracks overlaid on atmospheric flow patterns.

Figure 5.5: TECA was awarded the Juelich Supercomputing Center Prize for the Best application of HPC technologies for a pattern detection problem at the CAIP'15 conference.

### 5.2.2 Deep Learning for Science

In recent years, Deep Learning (DL) has enabled fundamental breakthroughs in computer vision, speech recognition and control system problems, thereby enabling a number of novel commercial applications. At their core, these applications solve classification and regression problems, tasks which are shared by numerous scientific domains. We assert that Deep Learning is poised to have a major impact on domain sciences, but there are unique challenges that need to be overcome first.

The primary challenge is in analyzing massive quantities of complex, multi-variate scientific data. Current Deep Learning implementations can take days to converge on O(10) GB datasets; contemporary scientific datasets are TBs-PBs in size. Scientific datasets often contain dozens of variables, which is in contrast to the small number of channels in images and audio data. Scientists need to be able to leverage parallel computational resources to get reasonable turnaround times for training Deep Neural Networks (DNNs). It is therefore imperative that DL software delivers good performance not only on a single node but also across a large number of nodes. We now elaborate on two scientific drivers that motivate our optimization and scaling efforts.

| | pixels | channels | #images | Volume |
|---|---|---|---|---|
| Climate | 768x768 | 16 | 0.4M | 15TB |

Table 5.2: Characteristics of datasets used in semi-supervised detection study.

#### 5.2.2.1 Semi-Supervised Learning for Climate

In this work, we consider the challenge of scaling Deep Learning architectures on massive CPU-based HPC systems. We consider the task of detection (or finding bounding boxes) for Tropical Cyclones, Atmospheric Rivers and Extra-Tropical Cyclones. The problem can be formulated as that of object recognition in images, the difference being that climate images have 16 or more 'channels', and their underlying statistics are quite different from natural images. Consequently, we cannot leverage pre-trained weights from contemporary networks such as VGG or AlexNet. Earlier work conducted by [129] demonstrates that convolutional architectures can solve the pattern classification task for cropped, centered image patches. In this work we develop a *unified, semi-supervised* architecture for handling all extreme weather patterns and develop a methodology for predicting bounding boxes. Most importantly, our method provides an opportunity to discover new weather patterns that might have few/no labeled examples.

While our exploration is conducted in the context of two concrete applications, we believe that our approach, and the resulting lessons learned, can be generalized to a much broader class of data analytics problems in science.

### 5.2.3 Current State of the Art

#### 5.2.3.1 Deep Learning on single node

The core computation in deep learning is dense linear algebra, specifically matrix multiply and convolution operations. These operations differ significantly from those usually found in HPC applications; both in terms of shapes/sizes of the operands (typically tall skinny matrices), as well as domain specific features such as convolutions with multiple small filters. Hence, specific deep learning libraries have been developed for hardware such as NVIDIA GPUs [39] and CPUs [90, 76].

The hardware efficiency of these kernels heavily depends on input data sizes and model parameters (weight matrix dimensions, number of convolutions, convolution strides, padding, etc). DeepBench [36] is a recently developed benchmark from Baidu that captures best known performance of deep learning kernels with varied input sizes and model parameters on NVIDIA GPUs and Intel® Xeon Phi™[1]. Their results show that while performance can be as high as 75-80% of peak flops for some kernels, decreasing minibatch size (dimension 'N' for matrix multiply and convolutions) results in significant efficiency drops to as low as

[1]Intel, Xeon and Intel Xeon Phi are trademarks of Intel Corporation in the U.S. and/or other countries.

20-30% (at minibatch sizes of 4-16) on all architectures. As we shall see, this has implications on performance at scale.

#### 5.2.3.2 Deep Learning on multiple nodes

There have been many attempts to scale deep learning models across a cluster of nodes [4, 35, 83, 34]. In this work, we focus on scaling the training of a *single model* across a cluster as opposed to the embarassingly parallel problem of training independent models [215]. We discuss two common architectures, shown in Figure 5.6.

**5.2.3.2.1 Synchronous-parallel architectures** such systems use synchronization barriers and force computational nodes to perform iterations in lock-step. Typically, data parallelism is used where different nodes split a big mini-batch of samples, each processing a chunk of the data. Recent papers that have attempted to scale synchronous deep learning have stopped at a few hundred nodes [34, 83, 5]. Aside from communication there are other factors that limit synchronous scaling:

**Batch size** Most systems use some variant of SGD with batch sizes that range from 64 to 1024. Large batch sizes have been shown to cause slowdown in convergence [69], and degrade the generalization properties of the trained model [102]. The batch size is a limit on the number of nodes in data-parallel synchronous systems.

**Stragglers** Since a synchronization barrier is used, the duration of the iteration depends on the slowest node. Variability in the computation needed per sample, OS jitter and, importantly, variations in the throughput and latency in the interconnect lead to significant load imbalance. This effect gets worse with scale.

**5.2.3.2.2 Asynchronous and hybrid architectures** Conceptually, asynchronous architectures [242, 158] remove the synchronization barriers. Each node works on its own iteration (mini-batch) and produces independent updates to the model. Those updates are sent to a central parameter store, the *parameter server* (PS), illustrated in Figure 5.6. The PS applies the updates to the model in the order they are received, and sends back the updated model to the worker where the update originated.

Asynchronous systems do not suffer from straggler effects and are not limited by the total batch size in the same way that synchronous systems are, an important property at scale. Asynchronous methods are known to give significant computational benefits in large-scale systems [35, 23]. Recent work [146] sheds new light on the convergence properties of such systems and shows the importance of momentum tuning for convergence.

**Performance trade-off** The main side-effect of asynchrony is the use of out-of-date gradients: each update is computed based on an older version of the model and then sent

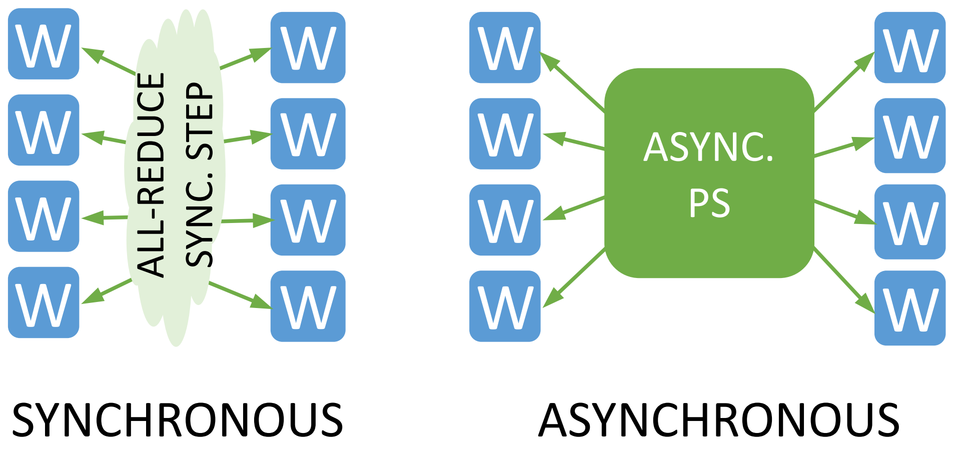


Figure 5.6: Schematic for synchronous and asynchronous communication schemes for updating parameters in distributed Deep Learning.

to the PS to be applied on the latest model. The number of updates that other workers perform between the time a worker reads the model and the time it sends its own update to the PS is called *staleness*. Asynchronous systems may need more iterations to solution, due to staleness: we say they have worse *statistical efficiency* [278, 69]. Synchronous systems typically take longer per iteration due to the straggler effect: they have worse *hardware efficiency*.

**Hybrid architectures** The trade-off between statistical efficiency vs. hardware efficiency suggests a third kind of architecture: a *hybrid* system [69]. In this architecture, worker nodes coalesce into separate *compute groups*. Each compute group follows a synchronous architecture: the workers split a mini-batch among themselves and produce a single update to the model. There is no synchronization across compute groups. A parameter server (PS) holds the model, and each compute group communicates its updates to the PS asynchronously. Given a cluster of fixed size, the number of compute groups (and their size) is a knob that controls the amount of asynchrony in the system. We can tune the amount of asynchrony along with the other hyper-parameters to find the optimal configuration. We use this hybrid architecture in our paper, as described in 5.2.4.4.

## 5.2.4 Innovations

### 5.2.4.1 Climate architecture

We formulate the climate problem as semi-supervised bounding box regression adapted from [189], which is inspired by [202, 128, 206]. Essentially, we have a fully supervised convolutional network for bounding box regression and an unsupervised convolutional autoencoder.

| Architecture | Input | Layer details | Output | Parameters size |
|---|---|---|---|---|
| Semi-supervised Climate | 768x768x16 | 9xconv 5xDeconv | coordinates, class confidence | 302.1 MiB |

Table 5.3: Specification of DNN semi-supervised architectures used in this study.

These two networks share various layers, so the extra unlabelled data input to the autoencoder can help improve the bounding box regression task. We use a series of strided convolutions to learn coarse, downsampled features of the input climate simulations. We call this series of convolutions the encoder of the network. At every location in the features, we compute 4 scores (confidence, class, x and y position of bottom left corner of box, and height and width of box) using a convolution layer for each score. At inference time, we keep only the boxes corresponding to confidences greater than 0.8. For the unsupervised part of our architecture, we use the same encoder layers but use the coarse features as input to a series of deconvolutional layers, which we call the deocoder. The decoder attempts to reconstruct the input climate image from the coarse features. The objective function attempts to simultaneously minimize the confidence of areas without a box, maximize those with a box, maximize the the probability of the correct class for areas with a box, minimize the scale and location offset of the predicted box to the real box, and minimize the reconstruction error of the autoencoder. As a solver, we use stochastic gradient descent with momentum.

#### 5.2.4.2 Single-node performance on manycore architectures

In this work, we used the Intel distribution of Caffe [86] to train our models. This distribution links in the Intel MKL 2017 library [90] with optimized deep learning primitives for Intel Xeon Phi. For our semi-supervised climate network, we needed optimized implementations of deconvolution that were not available. We used the fact that the convolutions in the backward pass can be used to compute the deconvolutions of the forward pass and vice-versa in order to develop optimized deconvolution implementations. These layers perform very similarly to the corresponding convolution layers.

#### 5.2.4.3 Multi-node scaling with synchronous approach

We utilize the new Intel® Machine Learning Scalability Library (MLSL) [87] for our multi-node implementation. This handles all communication required to perform training in a synchronous setting and enables different forms of parallelism - both data and model parallelism - to be applied to different layers of the network without the user/developer worrying about communication details. In this work, we deal with either fully convolutional networks or those with very small fully connected layers, so we only use data parallelism which is well suited for such layers. MLSL also introduces performance improvements over vanilla MPI implementations using endpoints - proxy threads/processes which drive communication on

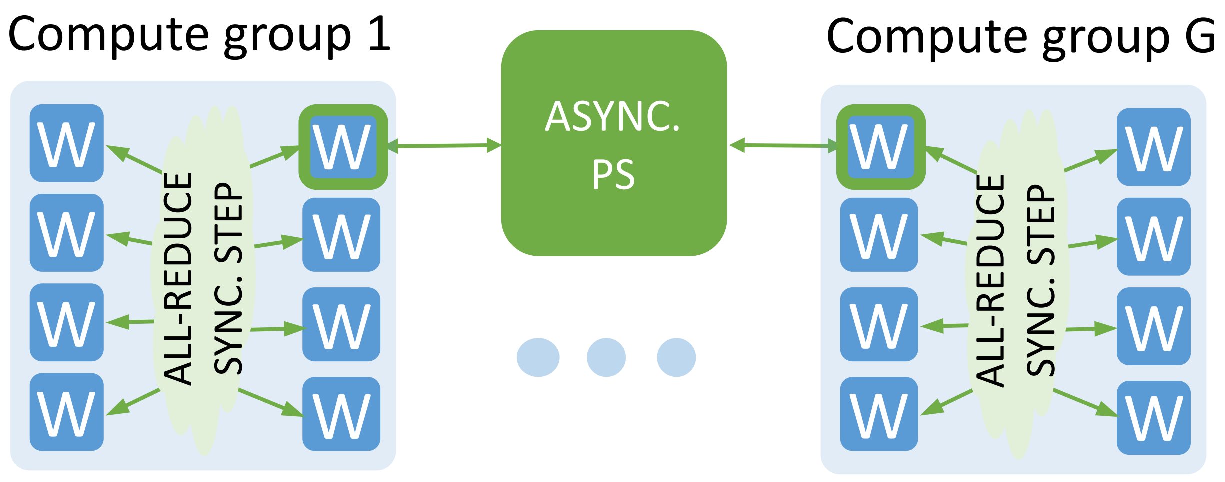


Figure 5.7: Novel hybrid architecture developed for scaling to massive concurrency on Cori/KNL CPUs.

behalf of the MPI rank and enable better utilization of network bandwidth. Results with this library have not been reported at large scales of more than a few hundred nodes; in this work, we attempt to scale this out to thousands of nodes.

#### 5.2.4.4 Multi-node scaling with hybrid approach

In 5.2.3.2.2 we outlined the limitations of fully synchronous systems that motivate asynchronous architectures. Asynchronous systems are not limited by the total batch size in the same way that synchronous systems are. Furthermore, asynchrony provides an added layer of resilience to node failures and the straggler effect. In this section, we describe the hybrid architecture we use in our system and discuss some of its novel elements.

Our architecture is inspired by recently proposed hybrid approaches [69], depicted in Figure 5.7. Nodes are organized into compute groups. Parallelization is synchronous within (using all-reduce) but asynchronous across groups via a set of parameter servers. The number and size of compute groups is a knob which controls the level of asynchrony and allows us to tune asynchrony and momentum jointly, as per recent theoretical guidelines [146]. Figure 5.8 shows an ideal placement of nodes and compute groups on Cori.[2] All-reduce operations are used to get the aggregate model update from all workers in the group. Then a single node per group, called the *root node*, is responsible for communicating the update to the parameter servers, receiving the new model and broadcasting it back to the group.

[2]For simplicity, PSs are shown in their own electrical group; however, this is not typically the case.

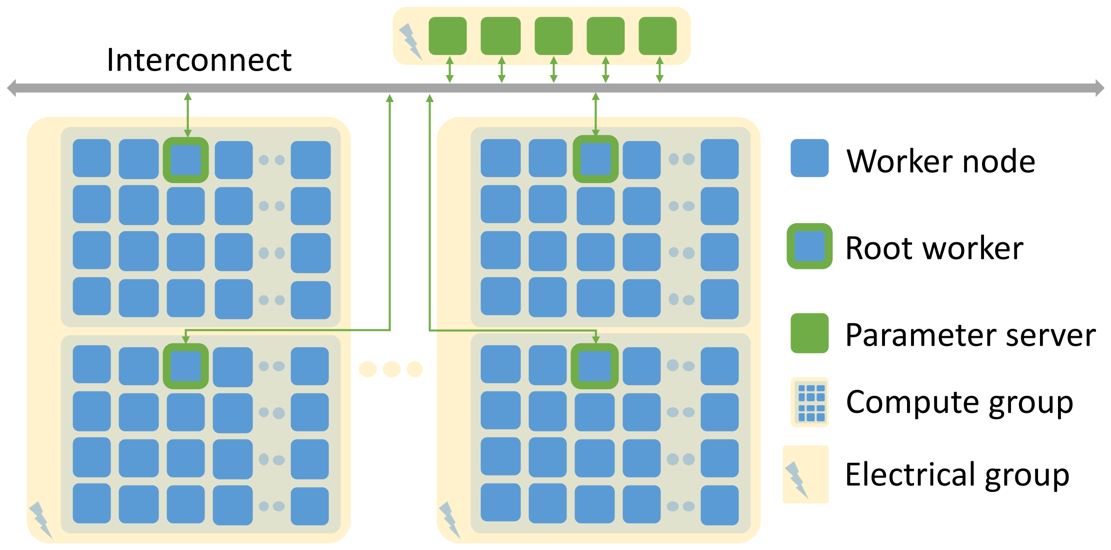


Figure 5.8: Topology-aware placement of worker nodes and parameter servers on the Cori/KNL system.

**Extreme Scale** Our work is the first instance of a hybrid architecture that scales to thousands of nodes. Previous implementations were designed (and typically deployed) on dozens or hundreds of commodity machines. For the present work, we deployed our implementation on configurations of up to 9600 nodes on an HPC system.

**Use of MLSL library** MLSL does not natively support asynchronous communication. Specifically, all nodes are assumed to communicate with each other, and the default library did not allow us to dedicate some subset of nodes for parameter servers. In this work, we extended MLSL to enable our hybrid implementation. Specifically, we extended MLSL to facilitate node placement into disjoint communication groups and dedicating nodes as parameter servers. Our new MLSL primitives allow for efficient overlaying of group communication and endpoint communication with the parameter server.

**Dedicated parameter servers for each layer** The parameter server needs to be able to handle the volume of network traffic and computation for the updates originating from multiple compute groups and for very large models. To reduce the chances of PS saturation, we dedicate a parameter server to each trainable layer in the network (Figure 5.9). We can consider each compute group as a bigger, more powerful node that performs the usual forward and backward pass operations on the layers of the network. The backward pass generates a gradient (model update) for each layer of the network. That update is communicated to its

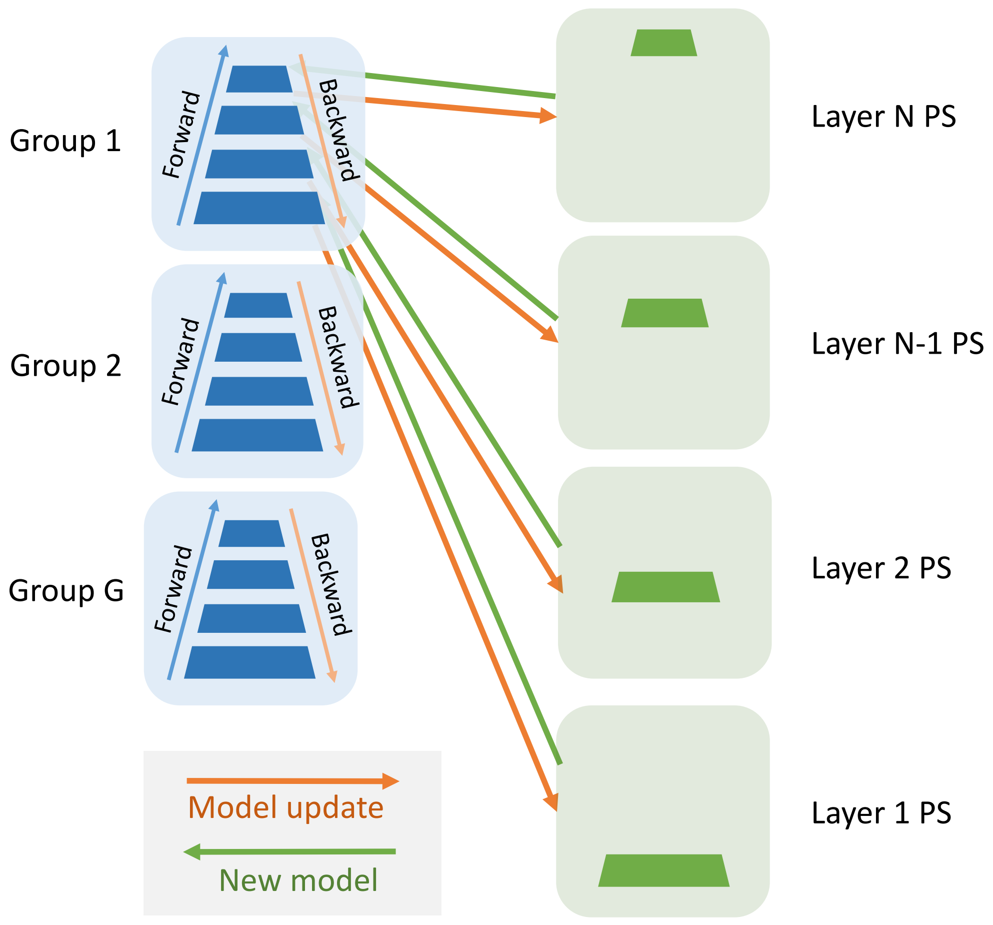


Figure 5.9: We assign a dedicated parameter server to each trainable layer of the network. Each group exchanges data with the PS for the corresponding layer. For clarity, we only depict the communication patterns for Group 1.

*dedicated parameter server*, the update is performed, and the model communicated back to the same compute group.

### 5.2.5 Cori Phase II

All experiments reported in this study are conducted on the Cori Phase II system at NERSC. Cori is a Cray XC40 supercomputer comprised of 9,688 self-hosted Intel Xeon Phi™ 7250 (Knight's Landing, KNL) compute nodes. Each KNL processor includes 68 cores running at 1.4GHz and capable of hosting 4 HyperThreads for a total of 272 threads per node.

The peak performance for single precision can be computed as: (9688 KNLs) x (68 Cores) x (1.4 GHz Clock Speed) x (64 FLOPs / Cycle) = 59 PetaFLOP/s. However, for sustained

AVX work, the clock-speed drops to 1.2 GHz, yielding a sustained peak performance of: 50.6 PetaFLOP/s.

Each out-of-order superscalar core has a private 32KiB L1 cache and two 512-bit wide vector processing units (supporting the AVX-512 instruction set[3]). Each pair of cores (a "tile") shares a 1MiB L2 cache, and each node has 96GiB of DDR4 memory and 16GiB of on-package high bandwidth (MCDRAM) memory. The MCDRAM memory can be configured into different modes, where the most interesting being *cache* mode in which the MCDRAM acts as a 16GiB L3 cache on DRAM. Additionally, MCDRAM can be configured in *flat* mode in which the user can address the MCDRAM as a second NUMA node. The on-chip directory can be configured into a number of modes, but in here we only consider *quad-cache* mode, i.e. all cores are in a single NUMA domain with MCDRAM acting as a cache on DDR4 main memory. Furthermore, Cori features the Cray Aries low-latency, high-bandwidth interconnect utilizing the dragonfly topology.

## 5.2.6 Performance Measurement

We count the executed FLOPs using Intel® Software Development Emulator (SDE) [88]. SDE distinguishes the precision of the FLOP operations and the actual executed FLOPs in the masked SIMD instructions of the code. We use SDE to count the executed single-precision flops in the computational kernels (i.e, the neural network layers) of a single node. Given that all the nodes execute these layers the same number of times and using the same problem size, we compute the total FLOPs by multiplying the single node FLOPs by the number of nodes. The counted FLOPs constitute the vast majority of the application's FLOP operations. The application time is spent in an iterative training loop, where the computation performed in each training iteration is the same. However, in some iterations, a checkpointing is performed to save the current trained model to the filesystem; this imposes some overhead on runtime. We measure the wall clock time per iteration to obtain the flop rate (i.e. iteration's measured FLOPS / iteration's time). The peak flop rate is obtained from the fastest iteration, while the sustained flop rate is computed from the best average iteration time in a contiguous window of iterations.

In the following section, we present the results of training the climate networks on the Intel Xeon Phi nodes of the Cori supercomputer. All our experiments use 66 of the 68 cores on each node, with 2 being reserved for the OS. All our experiments deal with single precision data and model parameters.

## 5.2.7 Performance Results

### 5.2.7.1 Single node performance

Figure 5.10 shows the flop rates and time spent in various layers for the Climate network. For a batch size of 8 images, the overall flop rate of the Climate network stands at 2.09

[3]This includes the subsets F, CD, ER, PF but not VL, BW, DQ, IFMA, VBMI.

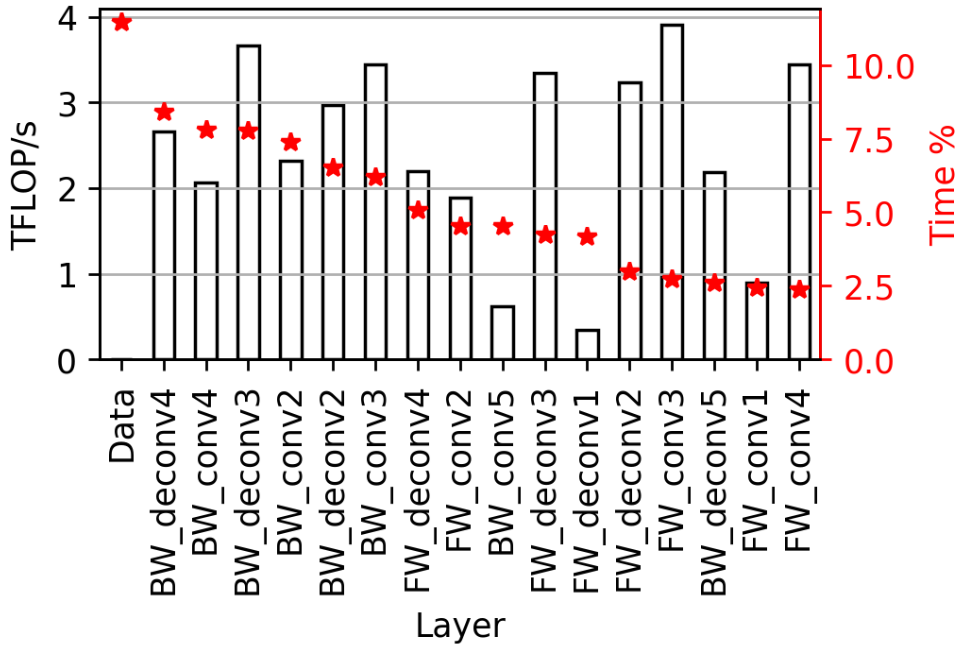


Figure 5.10: Single node runtime and flop rate of the top time consuming components, with batch size 8

TFLOP/s. Most of the runtime is spent in convolutional layers, which can obtain between 3.5 TFLOP/s for layers with many channels, and around 1.25 TFLOP/s on the initial layers with very few channels. As mentioned previously in DeepBench [36], the shapes of the parameters and inputs to a layer can affect performance significantly; we observe that in our experiments.

For the climate network, time spent in I/O (13%) for loading the data is significant; recall that climate problem consists of high resolution, 16-channel data. We have identified two bottlenecks in our current I/O configuration: First, I/O throughput from a single Xeon Phi core is relatively slow; second, the current HDF5 library is not multi-threaded. We will address these limitations in future work.

#### 5.2.7.2 Multi-node scaling

**5.2.7.2.1 Strong Scaling** The strong scaling configuration (involving keeping the overall batch size per update step fixed while varying the number of nodes) is a natural use-case for deep learning. Figure 5.11 shows strong scaling results for the climate network. We show 3 configurations: 1 synchronous group, 2 and 4 hybrid groups; and we show scalability from 1 to 1024 nodes. We use a batch size of 2048 per update. For the synchronous configuration, all nodes split the batch of 2048 images; for hybrid configurations, each compute group inde-

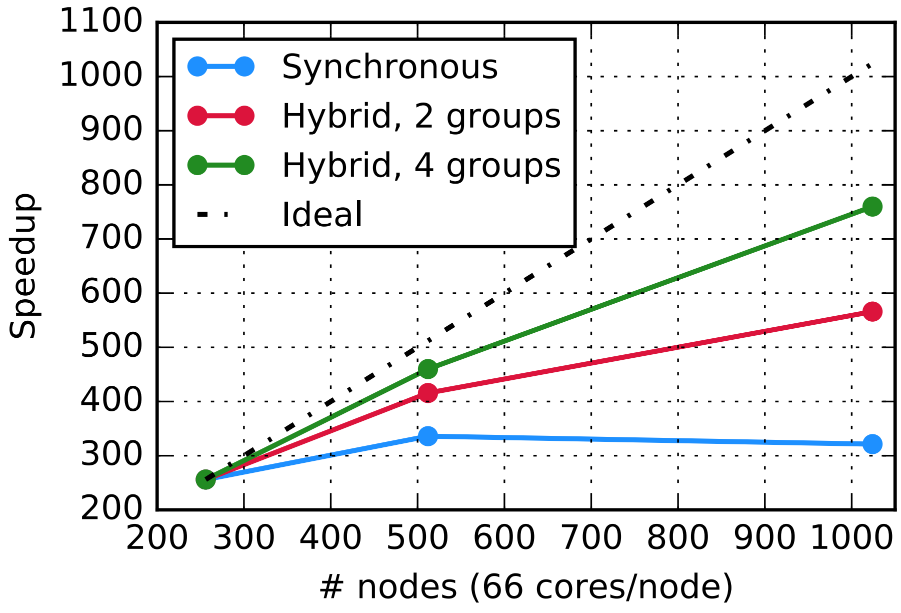


Figure 5.11: Strong scaling results for synchronous and hybrid approaches (batch size = 2048 per synchronous group).

pendently updates the model and is assigned a complete batch. Figure 5.11 shows that the synchronous algorithm does not scale past 256 nodes; 1024-node performance is somewhat worse than for 256. The scalability improves moderately for 2 hybrid groups, which saturates at 280x beyond 512 nodes, and more significantly with 4 hybrid groups, with about 580x scaling at 1024 nodes. We observe similar trends for the climate network in Figure 5.11 - the synchronous algorithm scales only to a maximum of 320x at 512 nodes and stops scaling beyond that point. The 2 and 4 group hybrid groups continue scaling to 1024 nodes, with scalability improving from 580x (on 1024 nodes) for 2 hybrid groups to 780x for 4 hybrid groups. There are two main reasons for this. One, in hybrid algorithms, only a subset of nodes need to synchronize at each time step, which reduces communication costs and straggler effects. Two, the minibatch size per node is higher for the hybrid approaches resulting in better single node performance. Scaling for our hybrid approaches is still not linear due to the single node performance drop from reduced minibatch sizes at scale.

**5.2.7.2.2 Weak Scaling** Figure 5.12 shows weak scaling for the climate network, where we keep a constant batch size (8 per node) across all configurations (synchronous and hybrid). The scaling results for the climate network are near-linear (1750x for synchronous and about 1850x for hybrid configurations).

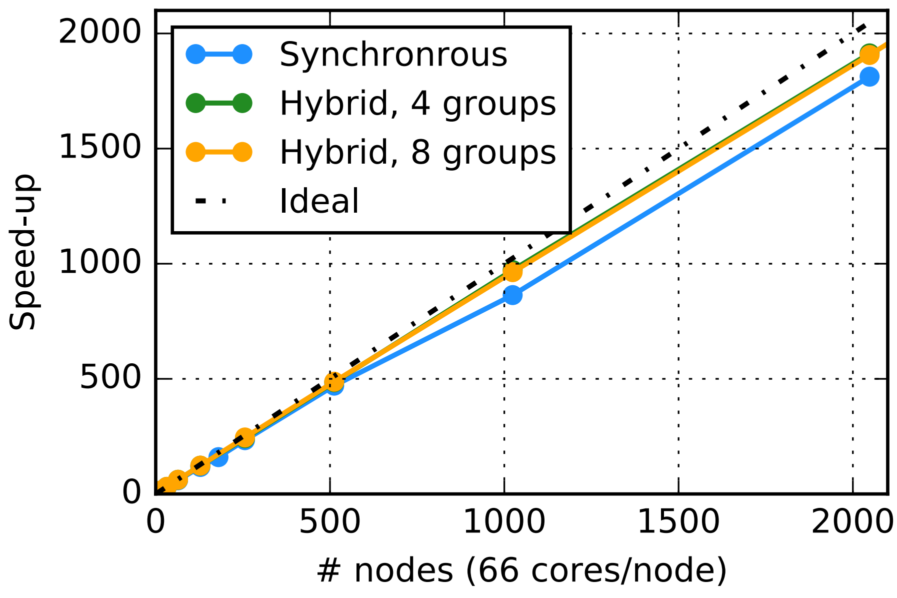


Figure 5.12: Weak scaling results for synchronous and hybrid approaches (batch size = 8 per node).

**5.2.7.2.3 Overall Performance** For the climate network, we obtained a **peak** throughput of **15.07 PFLOP/s** for a configuration of 9622 total nodes (9608 compute nodes plus 14 parameter servers) split into 8 groups, with each group using a minibatch of 9608. This corresponds to a speedup of 7205X over single node performance. The **sustained** throughput as measured over a 10 iteration span is about **13.27 PFLOP/s**, corresponding to a speedup of an average per-iteration runtime of 12.16 seconds. The sustained throughput computed includes the overhead of storing a model snapshot to disk once in 10 iterations, causing slowdowns.

## 5.2.8 Science Results

### 5.2.8.1 Climate Science Result

Figure 5.13 presents a sample image that illustrates the ability of our semi-supervised architecture to produce bounding boxes and class labels. In the figure, the architecture does a good job of localizing and identifying tropical cyclones. We are working on generating additional metrics for assessing the accuracy of bounding boxes for known classes (including extra-tropical cyclones and atmospheric rivers). More importantly, we are evaluating the ability of the architecture to discover *novel* weather patterns. Since this is a fundamentally new approach for pattern detection in the climate science community, we do not have a well-established benchmark to compare our results to.

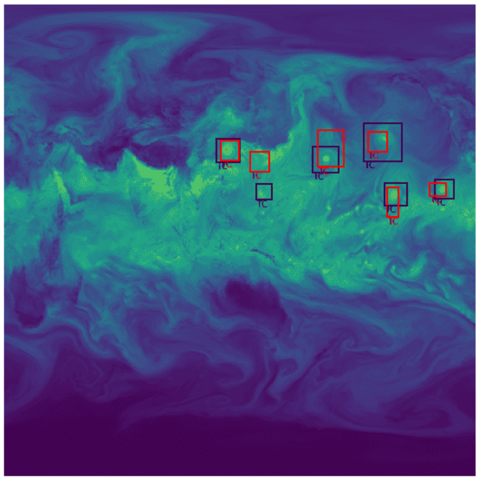


Figure 5.13: Results from plotting the network's most confident (>95%) box predictions on an image for integrated water vapor (TMQ) from the test set for the climate problem. Black bounding boxes show ground truth; red boxes are predictions by the network.

## 5.3 Exascale Deep Learning on GPUs

### 5.3.1 Collaborators

The research in this section was originally published in [115]. The collaborators were **Thorsten Kurth**, **Sean Treichler**, **Everett Phillips**, **Massimiliano Fatica**, **Joshua Romero**, **Ankur Mahesh**, **Mayur Mudigonda**, **Michael Matheson**, **Jack Deslippe**, and **Michael Houston**.

### 5.3.2 Overview

#### 5.3.2.1 Contributions

Motivated by the problem of finding extreme weather patterns in climate data, we make the following contributions:

- We adapt state-of-the-art Tiramisu and DeepLabv3+ architectures to solve segmentation problems on high resolution, multi-variate scientific datasets.

- We make a number of system-level innovations in data staging, efficient parallel I/O, and optimized networking collectives to enable our DL applications to scale to the largest GPU-based HPC systems in the world (Section 5.3.4.1).
- We make a number of algorithmic innovations to enable DL networks to converge at scale (Section 4.5.3).
- We demonstrate good scaling on up to 27360 GPUs, obtaining 999.0 PF/s sustained performance and a parallel efficiency of 90.7% (Section 5.3.6) for half precision. The peak performance we obtained at that concurrency and precision was 1.13 EF/s.
- Our code is implemented in TensorFlow and Horovod. Our performance optimizations are broadly applicable to the general deep learning + HPC community, and our stack is already being used by several other projects.

While our work is conducted in the context of a specific science driver, most of our proposed innovations are applicable to generic deep learning workloads at scale.

### 5.3.3 State of the Art

#### 5.3.3.1 State-of-the-art in Scientific Deep Learning

In recent years, the scientific community has begun to adopt deep learning methods and frameworks as tools for scientific analysis and discovery [181, 192, 140, 57, 52, 203]. Early applications were focused on adapting off-the-shelf convolutional neural networks from natural image processing applications or recurrent neural networks from speech recognition applications (for a review see [99]). There is currently a shift in the community towards incorporating scientific principles (e.g. physical laws such as energy or momentum conservation) and common assumptions (e.g. temporal and/or spatial coherence). Some recent examples in the domain areas related to ours include simulation of local wind field patterns via coupled autoencoder architectures [77], turbulence modeling for climate simulations via deep networks trained with loss functions that incorporate physical terms [253], and supervised applications of extreme weather pattern detection [129]. The field of physics-informed deep learning for scientific and engineering applications is in its infancy, and this section is a timely contribution focused on exploring the computational limits of representative architectures that many of the above approaches are based on.

#### 5.3.3.2 State of the Art in Large-Scale Deep Learning

Modern-day deep neural networks build upon the work laid out by McCullogh and Pitt [141], and Rosenblatt (perceptron) [208]. While forming the foundation for deep learning, these early models often struggled as the network size increases, limiting their utility in the analysis of complex systems. More recently, work by Krizhevsky [111] opened the flood gates for modern day Deep Learning, showing impressive performance on hard vision tasks using large supervised deep networks. This breakthrough was made possible in part by the rapid increase in computational power of modern computing systems. Since then, the complexity of tasks and the size of the networks have been growing steadily over the years, arguably requiring larger and more powerful platforms.

There has been more recent work on scaling deep learning up to larger node counts and performance. Preferred Networks, Inc. demonstrated ResNet-50 converging to 75% accuracy in 15 minutes using the ChainerMN [1] framework on 1024 NVIDIA Tesla P100 GPUs at a total global batch size of 32K for 90 epochs [2]. Jia at al. [96], concurrent with this work, demonstrated scaling to 2048 NVIDIA Tesla P40 GPUs at 64K batch size, achieving convergence in 6.6 minutes using TensorFlow. To effectively utilize leadership class systems, we need to push scaling significantly further than previous work. Most work on classification networks uses relatively small images from the computer vision community. Our work extends Deep Learning to handle much larger input in the form of snapshots from a scientific simulation. These "images" can be millions of pixels in size and generally have many more channels than the red, green, and blue of commodity imaging sensors. We are also contending with a significantly larger dataset that pushes the limits of the file system and requires new data handling techniques.

### 5.3.4 Innovations

#### 5.3.4.1 System Innovations

**5.3.4.1.1 High speed parallel data staging** Modern neural networks require large amounts of input data, and therefore training can easily be bottlenecked by an inability to bring input data to the GPU in a timely fashion. For instance, a single GPU training our modified Tiramisu network can consume 189 MB/s, already above the capabilities of a local hard drive, which means the 6 GPUs on a Summit node require a combined 1.14 GB/s. A training run using 1024 nodes therefore requires a sustained read bandwidth of 1.16 TB/s, and running on the full Summit system will require 5.23 TB/s, more than twice the target performance of the GPFS file system.

Summit makes available 800 GB of high-speed SSD storage on each node to help with local bandwidth needs. While a training data set can be quite large (the climate data used in this study is currently 3.5 TB), in a distributed training setting, it suffices for each node to have access to a significant fraction of the overall data set. The images selected by each rank are combined to form a batch, so a sufficient (and independently selected) set of samples for each rank to choose from results in batches that are statistically very similar to a batch selected from the entire data set. In our experiments, 250 images per GPU (1500 per node) are sufficient to maintain convergence.

Unfortunately, a naive staging script that asked each of 1024 nodes to copy its own subset of the full data set from GPFS required 10-20 minutes to complete and rendered the global file system nearly unusable for other users of the machine during that time. With this approach, each individual file from the data set was being read by 23 nodes on average. To address this, we developed a distributed data staging system that first divides the data set into disjoint pieces to be read by each rank. Each rank's I/O throughput was further improved by running multiple threads that perform file reads in parallel – using eight threads instead of one increased the achieved read bandwidth from 1.79 GB/s on average to 11.98 GB/s, an improvement of 6.7×. Once all files in the data set have been read from GPFS, point-to-point MPI messages are used to distribute copies of each file to other nodes that require it. This approach takes advantage of the significantly higher bandwidth of the Infiniband network and places no further load on the file system. Our improved script is able to stage in data for 1024 (4500) nodes on Summit in under 3 (7) minutes.

On Piz Daint, where no local SSDs are available, the only node local storage with sufficient bandwidth to feed the P100 GPU is the Linux `tmpfs` (DRAM), which has much more limited capacity.

**5.3.4.1.2 Optimized data ingestion pipeline** Although the staging of input data into fast local storage eliminates bottlenecks and variability from global file system reads, optimization is also required for the TensorFlow input pipeline that reads the input files and converts them into the tensors that are fed through the network. By default, the operations to read and transform input data are placed in the same operation graph as the networks themselves, causing idle time on the GPU while the CPU performs input-related tasks. This serialization can be eliminated by enabling the prefetching option of TensorFlow datasets, which allows the input pipeline to run ahead of rest of the network, placing processed input data into a queue. As long as the queue remains non-empty, the network can obtain its next input immediately upon completion of the previous one. The queue depth can be made deep enough to insulate against variability in the input processing rate, but the average production rate must still exceed the average consumption rate. As a further optimization, TensorFlow allows for concurrent processing of multiple input files using its `map` operator; however, the HDF5 library used to read the climate data serializes all operations, negating the benefit of parallel operation. By using the Python `multiprocessing` module, we were able to transform these parallel worker threads into parallel worker processes, each using its own instance of the HDF5 library. With 4 background processes taking care of reading and processing input data, the input pipeline can more closely match the training throughput of both networks, even when using FP16 precision.

**5.3.4.1.3 Hierarchical all-reduce** Network training is distributed across multiple GPUs using Horovod [218]. Horovod is a Python module that uses MPI to transform a single-process TensorFlow application into a data-parallel implementation. Each MPI rank creates its own identical copy of the TensorFlow operation graph. Horovod then inserts all-reduce operations into the back-propagation computation to average the computed gradients from each rank's network. Ranks update their local models independently, but (assuming consistent initialization) the use of gradients averaged across all the ranks results in identical updates (i.e. synchronous distributed training). Although it is possible for a TensorFlow+Horovod implementation to use multiple GPUs per rank, we adopted the simpler approach of using a different MPI rank for each GPU (i.e. 6 ranks per node on Summit), allowing the same code to be used on both Summit and Piz Daint. Horovod has been shown to have good scalability up to 1024 GPUs, but as we scaled further, we saw a dramatic loss in parallel efficiency resulting from two issues.

The first issue was a bottleneck on the first rank, which acts as a centralized scheduler for Horovod operations. As each TensorFlow process is independently scheduling the operations in its graph, different ranks might attempt to execute their all-reduce operations in different orders, resulting in deadlock. Horovod resolves this by dynamically reordering all-reduce operations to be consistent across all ranks. Each rank sends a message to the controller (rank 0) indicating readiness to perform a given all-reduce operation. Once the controller has received messages from all ranks for one or more operations, it sends a return message to every rank with an ordered list of tensors on which to perform collective operations. Our network has over a hundred allreduce operations per step, forcing the controller to receive and then send millions of messages per second for larger jobs. A distribution of the scheduling load is not possible, as all ranks must agree on a total order of collective operations to perform, so we chose instead to perform hierarchical aggregation of the control messages. The ranks are organized into a tree of configurable radix $r$, and each node in the tree waits for readiness messages from all of its direct children (and its own local operation) before sending a readiness message to its parent in the tree. Rank 0 sits at the root of the tree and uses the original Horovod algorithm for scheduling, but operates as if there were only $r+1$ ranks to coordinate. When a rank receives a message to start collective operations, it first relays that message to its children (if any) and then initiates the collective. This recursive broadcast approach guarantees that no rank sends or receives more than $r+1$ messages for each tensor, reducing the message load to mere thousands of messages per second, regardless of scale. Tuning of broadcast tree shapes can be important when latency is a concern, but TensorFlow's dynamic scheduler makes it fairly tolerant to small latency differences, and we observed no measureable performance difference for values of $r$ between 2 and 8.

The second issue to address was the performance of the collective all-reduce operations themselves. The existing Horovod implementation is able to reduce data residing on GPUs in two different ways, either by a standard `MPI_Allreduce` or by using the NVIDIA Collective Communications Library (NCCL)[163]. Both have their strengths: MPI often uses tree-based communication patterns for performance at scale, while NCCL uses a systolic ring approach that takes advantage of the bandwidth of GPUs that are connected with NVLink within a Summit node. To obtain both the scalability of MPI and the local bandwidth improvements of NCCL, we implemented a hybrid all-reduce approach. Data is first reduced across the GPUs within a node using NCCL. Once those 6 ranks have the same locally-reduced data, 4 of the ranks (two on each CPU socket) each perform an `MPI_Allreduce` on a quarter of the data, sharing with the corresponding rank on every other node and obtaining their quarter of the final result. Finally, NCCL broadcast operations are used within the node to ensure each of the 6 GPUs has a full copy of the entire all-reduce result. The decision to have 4 local ranks perform MPI operations was based on experimentation but suggests that a 1:1

mapping between communicating processes and virtual network devices is the most efficient strategy on Summit (each node has a dual-rail Mellanox IB ConnectX-5 EX adapter that is virtualized as 4 IB devices). With only a single GPU per node, Piz Daint does not benefit from this hybrid all-reduce implementation, but with the trend towards higher GPU counts per node, we expect this optimization to be beneficial on future machines as well.

### 5.3.5 Performance Measurement

| Network | Operation Count (TF/sample) | GPU Model (System) | Precision | Training Rate (samples/s) | Performance (TF/s) | % Peak |
|---|---|---|---|---|---|---|
| DeepLabv3+ | 14.41 | V100 (Summit) | FP16 | 2.67 | 38.45 | 31 |
| | | | FP32 | 0.87 | 12.53 | 80 |
| Tiramisu | 4.188 | V100 (Summit) | FP16 | 5.00 | 20.93 | 17 |
| | | | FP32 | 1.91 | 8.00 | 51 |
| | 3.703* | P100 (Piz Daint) | FP32 | 1.20 | 4.44 | 48 |

Figure 5.14: Single GPU performance results from training the Tiramisu and DeepLabv3+ networks. Results are shown for all tested systems using FP32 and FP16 precision where relevant. Note that the operation count for Tiramisu on Piz Daint (marked with an *) is computed from a modified network using 4 out of the 16 available input data channels.

Training performance of a deep neural network is generally reported in images (or batches) per second, but it can be useful to convert these numbers into floating point performance (i.e. FLOP/s). To do so, we incorporate some Python code that performs an analysis on the TensorFlow operation graph constructed by the application. The nodes of the graph are traversed, and the number of FLOPs required for each operation is computed. This graph-based analysis is essential for computing an accurate FLOP count when working with an application that defines multiple networks that share nodes.

For convolution nodes, additional analysis was required as there are multiple algorithmic formulations available, some of which require different quantities of floating point operations. TensorFlow dynamically tunes the algorithm choice for best performance, so it was necessary to use the API tracing capability in cuDNN to determine the algorithm selection. With the current versions of TensorFlow and cuDNN, we found that all convolutions were performed using either implicit GEMMs or direct convolutions. For example, a $3\times3$ direct convolution on a $1152\times768$ image with 48 input channels, 32 output channels and a batch size of 2 requires $3\times3\times1152\times768\times48\times32\times2\times2 = 48.9\times 10^9$ FLOPs. (The final factor of 2 follows the normal convention of counting both multiplies and additions as FLOPs.)

Once the FLOP count per step has been determined, we normalize this by the number of samples (images) per step. Based on the number of steps, we can then compute the number of samples processed in that step per rank and compute statistics on the time series of steps. If not otherwise stated, we compute the mean number of processed samples for every step over ranks and the median of the result over time and quote this as our sustained throughput.

We further compute an (asymmetric) error bar based on the central 68% confidence interval (computed from the 0.16 and 0.84 percentiles) over time. Using the FLOP per sample we can then compute a FLOP rate by multiplying the total processed samples per second with the FLOP per sample.

As is common for deep learning training situations, a series of additional calculations is carried out on the validation data set after each epoch, i.e. a full pass over the training data has been performed. Our data staging technique holds the number of steps in an epoch constant as we scale to larger node counts, keeping the epoch sizes large enough that this overhead is negligible once amortized over the steps.

#### 5.3.5.1 HPC Systems and Environment

**5.3.5.1.1 Piz Daint** Piz Daint at CSCS [180] is a hybrid Cray XC40/XC50 system. We will only consider the XC50 portion of the machine in this paper. The latter is comprised of 5320 hybrid CPU+GPU nodes. The CPU are single-socket Intel Xeon E5-2695v3 with 12 hardware cores which can host 2 HyperThreads each at 2.6 GHz. Each node has 64 GB of DDR memory and is further equipped with one NVIDIA Pascal GPU (P100) with 16 GB HBM2 memory and 32 GB/s PCIe bidirectional bandwidth. The nodes are connected by a low-latency high-bandwidth Aries interconnect with a diameter-5 Dragonfly topology. The peak single-precision floating point performance of the machine is 50.6 PF/s, twice the quoted 25.3 PF/s double-precision performance [179]. The global LUSTRE file system offers a peak bandwidth of 744 GB/s for reads and a total capacity of 28 PB.

*Software environment:* On Piz Daint, we use TensorFlow v1.6, compiled with CUDA 8.0 and the cuDNN 7.1.1 backend. We use our improved Horovod with hierarchical control plane which is based on the official v0.12.0 release. We compile it against Cray MPICH v7.6.0 and enable CUDA-aware collectives.

**5.3.5.1.2 Summit** Summit is the new leadership class supercomputer installed at the Oak Ridge National Laboratory (ORNL). This system is the current top-ranked supercomputer in the TOP500 rankings, the first on the list to surpass the 100 double-precision PF mark on the HPL benchmark [234]. The system is comprised of 4608 nodes, each equipped with two IBM Power 9 CPUs and 6 NVIDIA Volta GPUs (V100) with 16 GB HBM2 memory. Each Power 9 CPU is connected to 3 Volta GPUs using NVIDIA high-speed interconnect NVLink, capable of 300 GB/s bi-directional bandwidth. Each node has 512 GB of system memory and a 1.6 TB NVMe disk, half of which is available to jobs to be used as burst buffer. Dual-rail EDR Infiniband cards connect all the nodes using a non-blocking fat-tree topology. The nodes can access a POSIX-based IBM Spectrum Scale parallel file system with a current capacity of 3 PB and approximate maximum speed of 30 GB/s.

The Volta architecture includes Tensor Cores that provide mixed-precision operations. In each cycle, each of the 640 Tensor Cores can perform 64 floating-point Fused-Multiply-Add (FMA) operations with input values in half precision and output values either in half (FP16) or single precision (FP32). Deep Learning workloads are able to use mixed-precision.

Utilizing the Tensor Cores, each Volta GPU can perform 125 trillion floating-point operations per second, resulting in a peak node performance of 750 TF/s.

*Software environment:* On Summit, we use TensorFlow v1.8 compiled with CUDA 9.2 and cuDNN v7.2 backend. We again use our improved Horovod with hierarchical control plane which is based on the official v0.12.0 release. We compile it against IBM Spectrum MPI v10.2 and also NCCL v2.2.13 for fast GPU-based intranode all-reduces.

## 5.3.6 Performance Results

### 5.3.6.1 Single GPU Performance

Using the methodology described in Section 5.3.5, we determined the number of floating point operations required to process a single image with the Tiramisu and DeepLabv3+ networks. Combining these values with the sustained training rate (in samples/s) per GPU yields the sustained single GPU compute performance in Flop/s for each network. For both networks, a single image per GPU is processed per training step when FP32 precision is used, while for FP16, the lower memory footprint enables batches of two images per GPU to be processed during each training step. Single GPU performance results from this analysis can be found in Figure 5.14. From the tabulated data, we observed that the DeepLabv3+ network utilizes compute resources more efficiently than Tiramisu, achieving a higher percentage of peak Flop/s for both FP32 and FP16 computations. However, when comparing FP32 to FP16 computations across all results, the FP16 results are notably less efficient.

To determine the source of these performance inefficiencies, we made a detailed analysis of the work performed on the GPU using the CUDA profiling tools. Figure **??** provides a summary of this analysis, with further per-network details shown in Figure **??** (Tiramisu) and Figure **??** (DeepLabv3+). In order to capture the cost of all-reduce operations, this analysis was performed on a job running across 4 Summit nodes (24 GPUs), so the numbers differ slightly from the single GPU performance discussed above. Multiple runs were required to measure different performance counters, and the non-determinism in TensorFlow's execution of the graph adds some noise to the measurements (which, in some cases, causes ratios to slightly exceed 100%). Further, each training step requires thousands of kernels, making a traditional roofline analysis difficult. Instead, we grouped kernels into eight categories and looked at the computational and memory needs of each category. All of the computationally intensive kernels are in the forward and backward convolutions, which get fairly good utilization of the FP32 computing resources. However, the analysis shows that the Tiramisu network's convolution kernels become memory limited when using FP16 precision. This is a fundamental limitation of the Tiramisu-style network due to its small filter sizes per layer. The convolutions in the DeepLabv3+ use much larger channel counts per layer, resulting in higher computational intensity. This reduces the overall memory demand and improves datapath utilization. In addition to the convolutional layers, neural networks require many point-wise operations in the forward and backward passes (e.g. bias and dropout layers) as well as in the optimizer. The most expensive of these are in the forward pass, and get very good memory utilization for both FP32 and FP16 precisions. A small but significant contribution to the overall step time comes from copies and transpose operations that are inserted by TensorFlow. Although both networks can be implemented without extra copies (by assembling layers in place), the TensorFlow graph optimization pass is not able to make such a specific optimization. As a final optimization, we modified the data layout of the decoder stage of the DeepLapv3+ network to produce fewer extraneous transposes. This modification yielded a 10% speedup compared to the original code for our largest scale run. Finally, the NCCL kernels used for the intra-node portion of the all-reduce operations are bottlenecked by the bandwidth of the NVLink connections rather than the DRAM bandwidth (as described in 5.3.4.1.3, the MPI portion of the all-reduces is performed on the CPU concurrently with GPU and is not shown here).

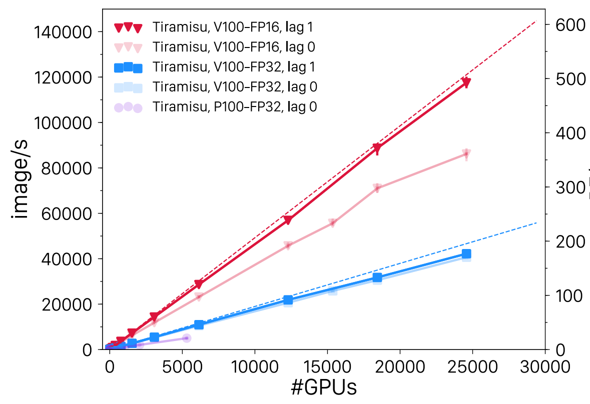
Our analysis found that the GPU is kept completely busy for the FP32 cases, indicating that any performance improvements have to come from optimizing or eliminating some of the kernels running on the GPU. The most beneficial kernels to optimize are the convolutions, but with so many different kernels being used, the effort would be significant and would deny the benefit of any improvements that are made to the cuDNN library. For example, a move from cuDNN v7.0.5 to v7.1.2 early in the project resulted in a 5% performance improvement with no changes to the application. We explored a move away from TensorFlow, implementing the network directly with cuDNN library calls, but the resulting code was much harder to maintain than the TensorFlow version. A 5-10% performance gain was not worth the impact on programmer productivity. The final optimization strategy, and the one we are pursuing, is to make incremental improvements within TensorFlow to improve the memory management and fuse some of the point-wise operations together to reduce the number of times tensors are read and written to DRAM. This might also allow the batch size to be increased, which would also improve the efficiency of the convolutional stages.

With the use of significantly faster math in the FP16 cases, the memory-bound kernels consume a larger fraction of the overall step time, and any optimizations to eliminate copies or fuse point-wise tasks will help the FP16 even more than FP32. The profile for the FP16 also shows some periods where the GPU has run out of work, suggesting that code running on the CPU such as the input pipeline or the TensorFlow scheduler may require additional optimization as well.

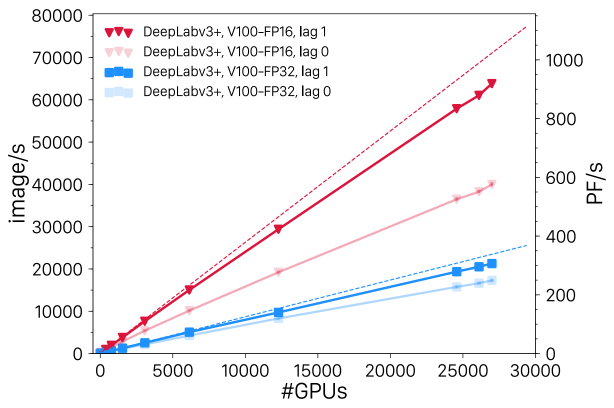
#### 5.3.6.2 Scaling Experiments

We perform several scaling experiments on Piz Daint and Summit. On Piz Daint, we ran the Tiramisu network only, while on Summit, both Tiramisu and DeepLabv3+ networks were run. The experiment setup is slightly different for the two systems, and we explain the details below. We bind one MPI rank to each GPU which amounts to one rank per node on Piz Daint and six ranks per node on Summit.

On Piz Daint, we scale the training up from a single GPU to the full machine, i.e. 5300 nodes. We also compare the scaling behavior when staging input data against reading it from the global Lustre file system. On Summit, we run with a single GPU as a baseline, but then sweep from 1 to 4560, nodes using all 6 GPUs per node (i.e. 6 to 27360, GPUs).



(a) Tiramisu



(b) DeepLabv3+

Figure 5.15: Weak scaling results in terms of images/sec and sustained performance in PF/s on Summit (FP16 and FP32, Tiramisu and DeepLabv3+) and Piz Daint (FP32, Tiramisu). The dashed lines represent the ideal scaling lines for the different architectures and precisions.

The scaling results are shown in Figure 5.15. We find that the training performance of Tiramisu scales to a sustained 21.0 PF/s on the full Piz Daint machine, achieving parallel efficiencies of 83.4% at 2048 nodes and 79.0% at 5300 nodes in FP32. On Summit, scaling Tiramisu to 4096 nodes yields a sustained throughput of 176.8 PF/s and 492.2 PF/s for FP32 and FP16 precision respectively, maintaining parallel efficiencies above 90% in both cases. Moving on to DeepLabv3+, scaling to 4560nodes with FP32 precision yields a sustained throughput of 325.8PF/s and a parallel efficiency of 90.7. The FP16 network reaches a peak 1.13 EF/s, sustained 999.0 PF/s and 90.7% parallel efficiency at that scale. The highest performing results were obtained on Summit in the cases with gradient lag (see 4.5.3.3) enabled, corresponding to the data labeled “lag 1" in Figure 5.15. The results clearly indicate the effectiveness of the lagged scheme in improving the overall application scalability.

To demonstrate the benefit of the data staging process described in 5.3.4.1.1, we experimented on Piz Daint with reading input data directly from the global file system and highlight results in Figure 5.16. Performance matches the runs using data staging at lower node counts, but the difference becomes apparent at larger scales. On 2048 GPUs, the parallel efficiency has dropped to 75.8%, a 9.5% penalty for not staging the input data in the local `tmpfs` storage. Additionally, the throughput shows larger variability. At this scale, the neural network is demanding nearly 110 GB/s of input data, very close to the file system’s limit of 112 GB/s. Therefore, we did not attempt to scale beyond 2048 nodes without data staging.

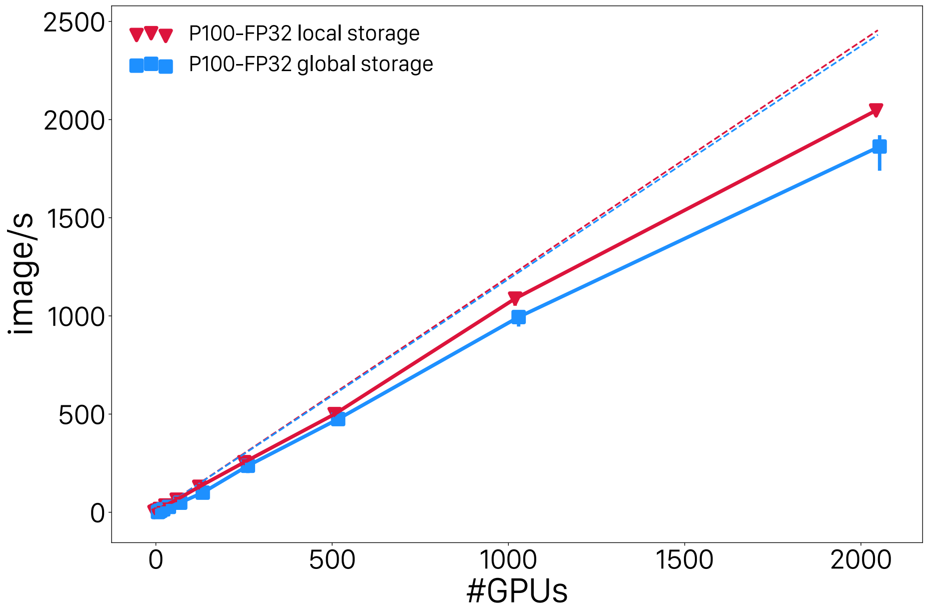


Figure 5.16: Dependence of weak scaling on input data location on Piz Daint.

#### 5.3.6.3 Convergence at Scale

A major challenge for deep neural networks is to maintain convergence properties (and with good accuracy) as the network training is scaled out. To demonstrate the stability of our network at large scales, we performed longer runs on up to 1024 Summit nodes using both the FP32 and FP16 precision modes, training the network to convergence. As with the scaling runs, the dataset is resampled to put 1500 files per node, improving the statistical properties of the large batches being used at this scale. The training in each case was performed for a fixed number of epochs (targeting a total training time of just over two hours).

The training loss curve for these runs are shown in Figure 5.17 along with curves for runs at smaller scales (384 GPUs and 1536 GPUs). Moving averages over 10 step windows are used to filter out step-to-step fluctuations in the loss. As can be seen in Figure 5.17, all of the configurations are converging with both FP16 and FP32. Tiramisu as well as DeepLabv3+ network is stable at large scale with the initially chosen set of hyperparameters. Tuning of hyperparameters is always necessary when scaling up a network, and we expect that the time to solution will improve further as they are dialed in. There are a few other important things to notice in Figure 5.17 : 1) FP16 converges in significantly less time than FP32; 2) DeepLabV3+ generally converges faster than Tiramisu; 3) and lag0 vs lag1 with DeepLabV3+ has nearly identical training loss curves. The ability to perform these experiments in an hour or two rather than days is a key enabler to being able to perform training at these scales and explore the hyperparameter and algorithm space.

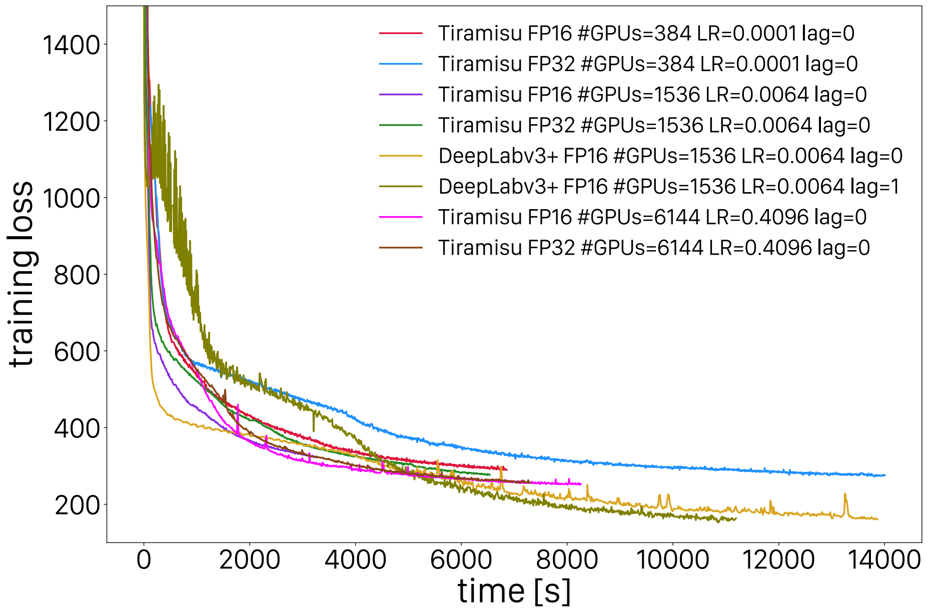


Figure 5.17: Training loss curves for various concurrencies and precisions for the Tiramisu and DeepLabv3+ architectures.

### 5.3.7 Gordon Bell Prize

This work was awarded the prestigious Gordon Bell Prize at ACM SuperComputing 2018. The Gordon Bell prize is the topmost honor in the field of High Performance Computing, and second only to the Turing Award in the field of Computer Science.

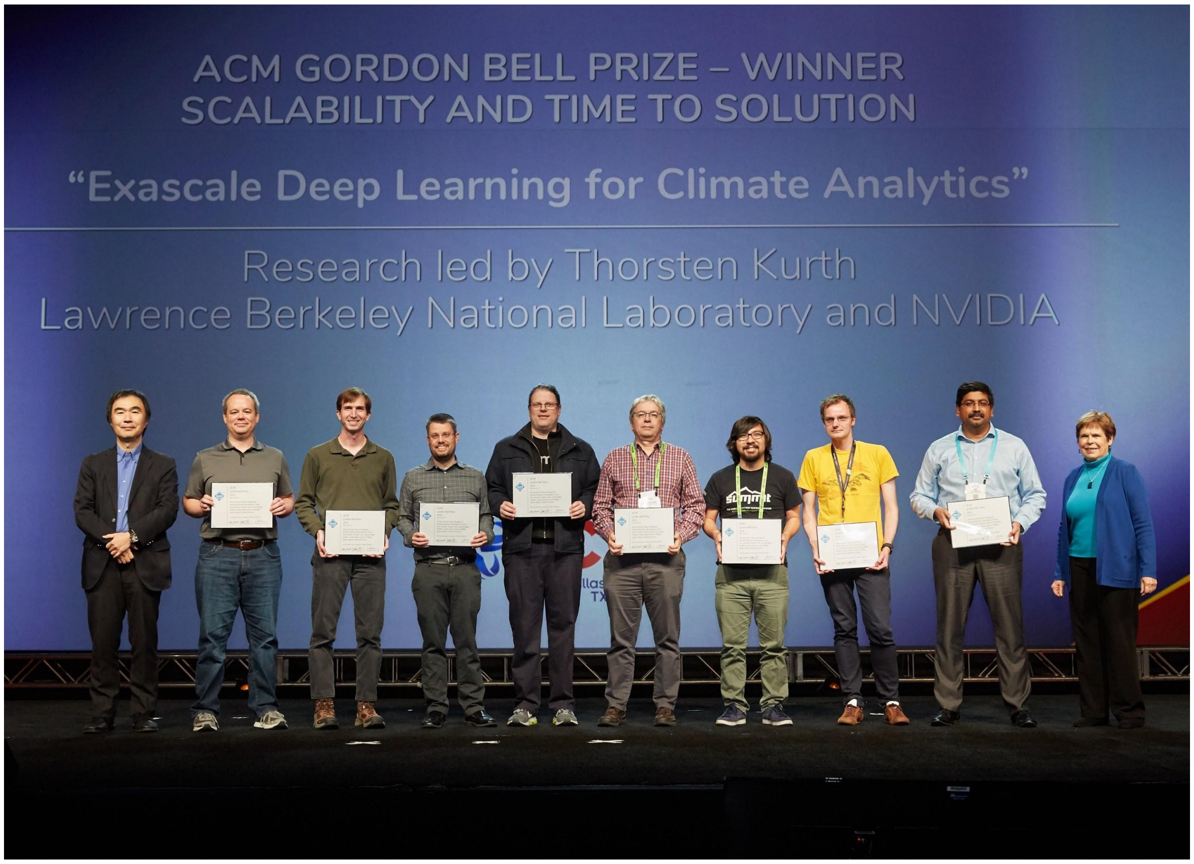

ACM GORDON BELL PRIZE – WINNER
SCALABILITY AND TIME TO SOLUTION
"Exascale Deep Learning for Climate Analytics"
Research led by Thorsten Kurth
Lawrence Berkeley National Laboratory and NVIDIA

Figure 5.18: Gordon Bell Prize was awarded to our team for developing the first exascale Deep Learning application in the world.

# Chapter 6

# ClimateNet

## 6.0.1 Collaborators

This chapter is from the paper [185]. The collaborators were **Karthik Kashinath**, **Mayur Mudigonda**, **Sol Kim**, **Lukas Kapp-Schwoerer**, **Ege Karaismailoglu**, **Andre Graubner**, **Leo von Kleist**, **Thorsten Kurth**, **Annette Greiner**, **Kevin Yang**, **Colby Lewis**, **Jiayi Chen**, **Andrew Lou**, **Sathyavat Chandran**, **Ben Toms**, **Will Chapman**, **Katherine Dagon**, **Christine A. Shields**, **Travis O'Brien**, **Michael Wehner**, and **William Collins**.

## 6.0.2 Motivation

To overcome long-standing challenges and discrepancies surrounding heuristics in the field of climate science, we have successfully adapted techniques from a different domain: namely Deep Learning (DL) from the field of computer science. The application areas of computer vision, speech recognition and robotics have struggled with custom heuristics since the mid-1980s, and have recently conclusively demonstrated that Deep Learning techniques can successfully and significantly advance the state of the art [124]; [126]. Inspired by these results, we have demonstrated that DL can indeed be applied to identifying the type (classification), spatial extent (localization), and pixel-level masks (segmentation) of weather and climate patterns [130, 80, 191, 113]. The success of these applications, however, was limited by the lack of high quality, reliable, labeled data which is a key requirement for the success of supervised DL techniques. The fields of weather and climate science currently lack these crucial datasets.

Scientists in both of these fields have been increasingly adopting the use of ML and DL, owing in part to the increase in available computational power and the ever growing volumes of data e.g. increases in temporal and spatial resolution of reanalysis products and climate models. ML and DL techniques, many of which were developed to work with 'big data', have shown demonstrated applications in meteorology and climate: parameterization in climate models, post-model bias correction, and forecasting of the El Niño Southern Oscillation

(ENSO) and Madden-Julian Oscillation (MJO) ([164]; [11]; [17]; [135]; [70]; [240]; [142]). To highlight one success, [70] demonstrated the skill of DL on forecasting El Niño states and found DL to forecast with superior leadtimes than state-of-the-art dynamical models. Many of the ML and DL techniques used, again rely on the availability and quality of labeled data. Some of the aforementioned papers utilize specific ENSO or MJO indices which have rigid and established definitions (e.g. Niño3.4) which allows for straightforward generation of labels (ENSO states) which can be used for training the DL model. However, even these large-scale modes have variety in their definition ([160]). One bottleneck to expanding the success of DL to a greater variety of weather and/or climate phenomena lies in the lack of large labeled datasets.

A major contribution of this thesis is the creation of 'ClimateNet' – an open, community-sourced human expert-labeled curated dataset – that captures tropical cyclones (TCs) and atmospheric rivers (ARs) in high-resolution climate model output from a simulation of a recent historical period. We use the curated ClimateNet dataset to train a state-of-the-art DL model for pixel-level identification, *i.e.* segmentation, of TCs and ARs. We then apply the trained DL model to historical and climate change scenarios simulated by the Community Atmospheric Model (CAM5.1) and show that the DL model accurately segments the data into TCs, ARs or 'the background' at a pixel level. Further, we show how the segmentation results can be used to conduct spatially and temporally precise analytics by quantifying distributions of extreme precipitation conditioned on event types (TC or AR) at regional scales. The key contribution of this work is that it paves the way for DL-based automated, hi-fidelity and highly precise analytics of climate data using a curated expert-labelled dataset – ClimateNet.

ClimateNet and the DL-based segmentation method provide several unique capabilities: (i) they can be used to calculate a variety of TC and AR statistics at a fine-grained level; (ii) they can be applied to different climate scenarios and different datasets without tuning as they do not rely on threshold conditions; and (iii) the proposed DL method is suitable for rapidly analyzing large amounts of climate model output. While our study has been conducted for two important extreme weather patterns (TCs and ARs) in simulation datasets, we believe that this methodology can be applied to a much broader class of patterns, and applied to observational and reanalysis data products via transfer learning.

## 6.1 Creating the ClimateNet Dataset

The first step towards building an expert-labeled dataset is the development of a labeling interface, whereby climate data can be ingested and climate experts can annotate events of interest, such as atmospheric rivers and tropical cyclones. The requirements for such an interface are: (i) sufficient information to annotate events correctly; (ii) ability to add, delete and modify labels easily, and (iii) facility to specify user confidence for each label individually.

### 6.1.1 ClimateContours

We develop the ClimateContours tool, which is a guided user interface for annotating climate events. ClimateContours is built upon the annotation tool LabelMe [212], which was originally developed to aid the generation of annotated examples for training supervised learning models in the computer vision community. ClimateContours is a versatile and easy-to-use tool, hosted at `http://labelmegold.services.nersc.gov/climatecontours_gold/tool.html`, which leverages the science gateway infrastructure at the National Energy Research Supercomputer Center (NERSC). ClimateContours renders snapshots from a prescribed climate dataset and allows the user to label two types of events - Atmospheric Rivers (ARs) and Tropical Cyclones (TCs). The labeler chooses the pen-like tool to manually place vertices of a polygon around an event of choice. The placement of vertices ceases when a convex hull is created, *i.e.* when the last vertex coincides with the first vertex. The labeler then chooses the type of event (AR, TC) and the confidence of their labeling process (high, medium, and low).

The labeler has the option to delete edges or the entire polygon and re-create polygons as many number of times as they wish. In addition, a labeler may zoom in to view events at a finer scale and switch between various views of raw and derived variables to help inform their labeling.

Currently, ClimateContours renders snapshots from high-resolution 25-km CAM5.1 climate model output [264]. This dataset contains dozens of physical variables of potential interest to weather and climate scientists; for example, wind velocity, temperature, pressure, and humidity at different vertical levels and across the globe. These variables contain information relevant to the dynamics of weather and climate phenomena, but not all variables are needed to detect a weather event. Based on the experience and wealth of knowledge accumulated by meteorologists and climate scientists, and for relative ease of use, we provide a subset of six variables - in various combinations - to the user to aid them in creating labels for TCs and ARs through ClimateContours. The variables are shown in the following table:

| Variable | Units |
|---|---|
| Integrated Vapor Transport | $kg\,m^{-1}\,s^{-1}$ |
| Integrated Water Vapor | $mm$ |
| Vorticity | $s^{-1}$ |
| Surface Wind Vectors | $m\,s^{-1}$ |
| 850 hPa Wind Vectors | $m\,s^{-1}$ |
| Sea Level Pressure | $hPa$ |

### 6.1.2 Labeling Campaigns

In order to capture the expertise of climate scientists in characterizing ARs and TCs, and to obtain sufficient data to train deep neural networks, we conducted multiple labeling campaigns across several institutions and events. These included campaigns at LBNL, UC Berke-

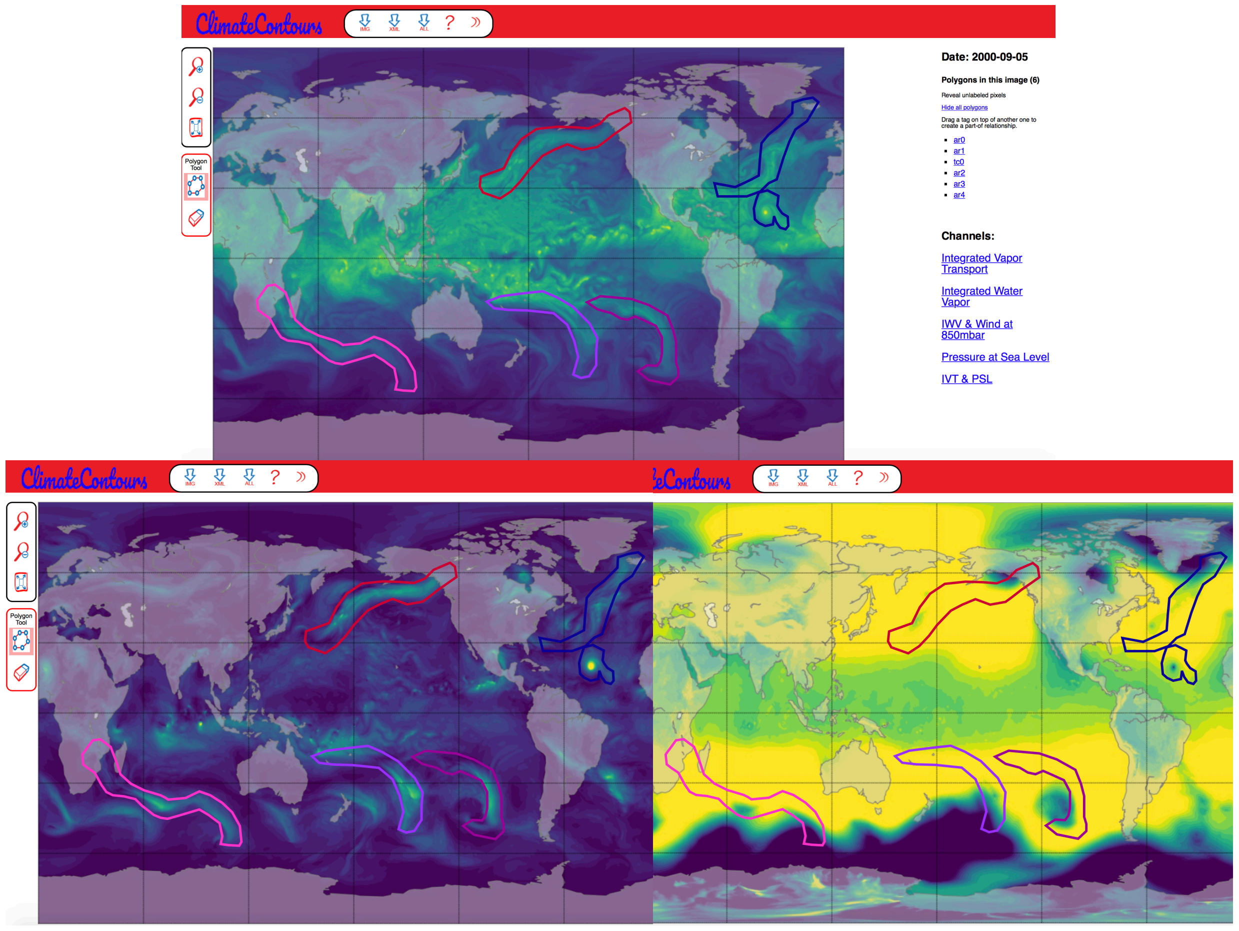


Figure 6.1: The ClimateContours web-based labeling interface. Labelers can choose different channels (physical variables) on the right side of the GUI to display different variables or combinations of variables on the global map. On top: Integrated Water Vapor (IWV) is shown with labels of ARs and TCs; bottom left: Integrated vapor transport (IVT); bottom right: pressure at sea level (PSL).

ley, NCAR, Scripps/UCSD, the 2019 ARTMIP Workshop and the 2019 Climate Informatics Workshop. For each labeling campaign, participants were briefed on how to use the ClimateContours tool and provided some background on the specifics of ARs and TCs and how to label them effectively. Overall, approximately 80 weather and climate scientists participated in the campaigns and contributed several hundred labelled snapshots of climate data. The ClimateNet dataset currently contains over 1000 carefully curated data labeled by experts using the ClimateContours tool. The labeling campaigns proved to be invaluable for not only for generating high-quality labeled data, but also for obtaining feedback on the ClimateContours tool itself, variables of interest, and how the labeling process could be improved.

### 6.1.3 Diversity of Expert Labels

Just as there exists a dozen different heuristics for detecting weather events such as atmospheric rivers, tropical cyclones and extra-tropical cyclones [221, 251, 157], we find differences in the labels provided by experts using the ClimateContours tool. This is perhaps not unexpected as different experts inherently conceptualize and identify weather events in slightly different ways. The labeling campaigns shed useful light on the diversity of labeling styles and implicit assumptions of different experts, as is seen in 6.2. Disagreements and disparities were most common on the exact spatial extents of individual storms, and less so on the presence or absence of the storm. Some experts disagreed on edge cases, *viz.* incipient events or those that were dissipating. However, we note that the disagreements between human labels was less severe than differences noticed in dedicated heuristic-based event detection intercomparison projects such as ARTMIP, TCMIP and IMILAST, which exhibit significant disparities in the presence or absence of labeled extreme events and their boundaries [221, 265, 244, 157, 251].

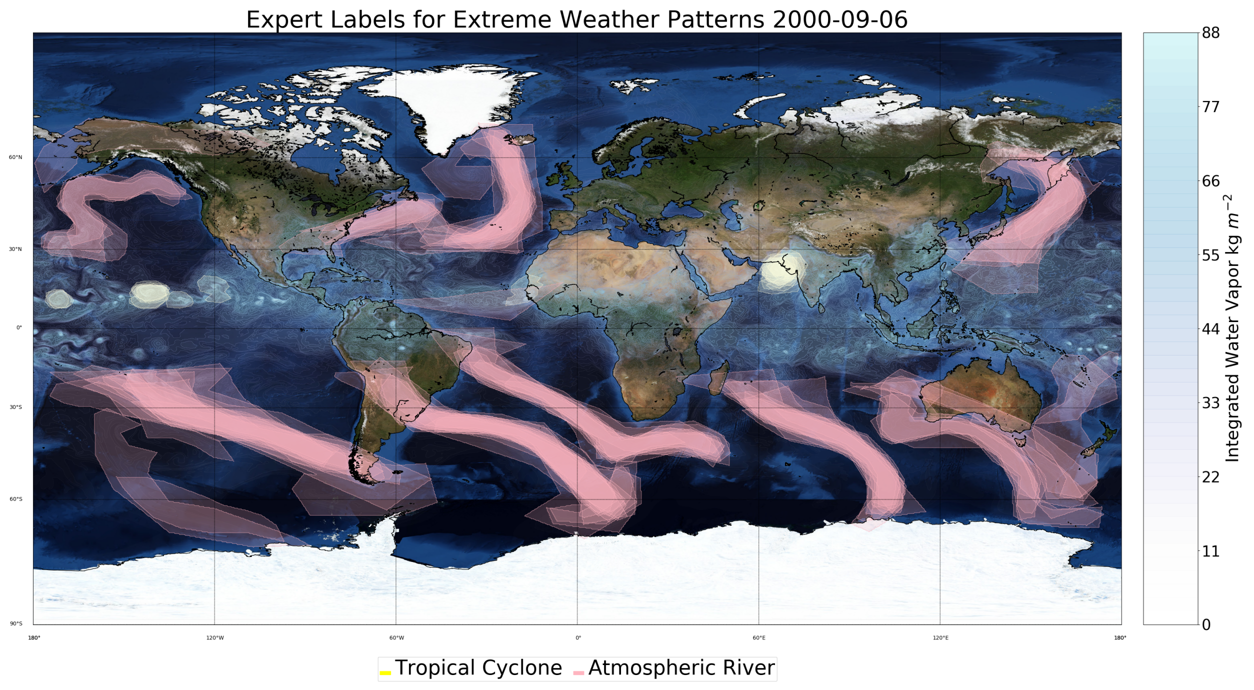


Figure 6.2: Comparison of 15 different expert labelings. Density of pink masks show overlap of AR labels, density of yellow masks show overlap of TC labels.

In Figure 6.2, labels from 15 different experts are shown. Most experts agree on some of the prominent ARs and TCs, albeit, with some variance in the precise boundaries. The two ARs in the southern Atlantic Ocean and the TC off the west coast of India are examples of strong expert agreement. However, there are also quite a few discrepancies. A few labelers considered there to be ARs off the east coast of Australia while most did not consider these patterns to represent ARs. Some of the smaller cyclonic structures in the equatorial Pacific also demonstrate discrepancies. One egregious error in labeling can be seen from the triangular AR polygon sitting on the equator in the Atlantic. Note that the ClimateContours interface does include the capability for labelers to specify masks using a free-form tool, as opposed to specifying polygonal shapes. We intend to expose and leverage this capability in future labeling campaigns.

### 6.1.4 Quality Assurance and Quality Control

Any manual labeling campaign, even one conducted amongst experts, is subject to a errors stemming from various sources: human errors (lack of expertise/understanding, lack of motivation/thoroughness, mis-interpretation of instructions, mislabeled events, missed events, fatigue) and technical errors (glitches in the web interface, infrastructure). It is simply unrealistic for us to expect that all images will be labeled to a consistently high degree of accuracy. In order to address this important issue, we formed a small team of QA/QC experts from the co-author list on this paper. The experts had a background in both climate and computer science; had a good working knowledge of TC and AR patterns, and were briefed on, and motivated to reach a high target accuracy for the labeled dataset. This core team manually examined and executed a thorough "Quality Assurance (QA) / Quality Control (QC)" processes on about 600 samples to correct for errors.

The top priority for the QA/QC team was to fix mis-labeled and missed events. A second type of QA/QC task was to modify the boundaries of correctly labeled events to be closer to the true shape, size and structure of ARs and TCs. The QA/QC team reached an internal consensus on basic defining characteristics such as: (i) TCs exist in the tropics and are sufficiently intense, measured by low sea level pressure and high voriticity, (ii) ARs sometimes are associated with extra-tropical cyclones (ETCs) but the ETCs should not be included in the AR boundary, (iii) slight differences exist in AR signatures in IWV and IVT fields, and we choose boundaries based on geometric criteria, *i.e. ARs are long, narrow, and transient corridors of strong horizontal water vapor transport ...*(`http://glossary.ametsoc.org/wiki/Atmospheric_river`).

Despite making such QA/QC adjustments to experts' labels, there remained some variety amongst AR and TC labels, perhaps representative of the lack of a clear theoretical and quantitative definition for these events. We argue that these relatively minor differences are not a detriment to the training and evaluation of the DL model, as will be shown in the results section next.

## 6.2 Applying Deep Learning on the ClimateNet Dataset

### 6.2.1 Deep Learning for Segmentation

In this section, we present our deep learning approach to generate high-quality segmentation masks (i.e. separating objects of interest from the background) for ARs and TCs, using the curated ClimateNet dataset. We model this problem as a semantic segmentation task, i.e. the goal is to assign a class label to every pixel for a given input image. In our case, the input image is the CAM5.1 25-km grid comprising of atmospheric fields, and the output class labels are TC, AR and background (BG).

#### 6.2.1.1 Model

We use a PyTorch implementation of DeepLabv3+ (*https://github.com/MLearing/Pytorch-DeepLab-v3-plus*). The input to the model consists of an array of size $(4, 1152, 768)$. It contains atmospheric data from four different channels; namely: TMQ (Total vertically integrated precipitable water), U850 (Zonal wind at 850 mbar pressure surface), V850 (Meridional wind at 850 mbar pressure surface) and PRECT (total convective and large-scale precipitation rate). The deep neural network architecture used in this work is the DeepLabv3+ architecture (see Figure 6.3) developed by [22] based off of [24]. This architecture has attained state-of-the-art results across various semantic segmentation benchmarks in the computer vision community (PASCAL VOC 2012 and Cityscapes). DeepLabv3+ consists of an *encoder* which captures rich semantic information across multiple scales. The idea of an *encoder* is a series of learnt, hierarchical filters that extracts useful information as pertaining to the task defined - in this case, segmenting ARs and TCs from background. The *decoder* module then upsamples or in other words, goes from a lower to higher resolution to produce refined object boundaries. There are learnable weights associated with both the *encoder* and the *decoder* which are learnt together while trying to minimize the loss. The loss for the network we use is essentially the $L_2$ difference between the predicted masks and the ground truth labels for a given input.

The output, as discussed earlier, is a segmentation mask of size $(1152, 768)$, where each element in the mask takes the value of either 0 (BG), 1 (TC) or 2 (AR).

#### 6.2.1.2 Training

We study the learning capabilities of DeepLabv3+ on two different datasets $D_1$ and $D_2$, which are derived from the CAM5.1 All-Hist configuration described in previous Sections. Dataset $D_1$ consists of 128k samples, where each sample conforms to the input format described above. The samples are from the time interval between January 1996 and February 2012, and are generated by the Community Atmospheric Model (CAM5.1) [264]. Every sample in this dataset has TCs and ARs heuristically labeled by the TECA toolkit [182]. For training, we split $D_1$ into a training set, which is a randomly sampled subset that contains 51.2k

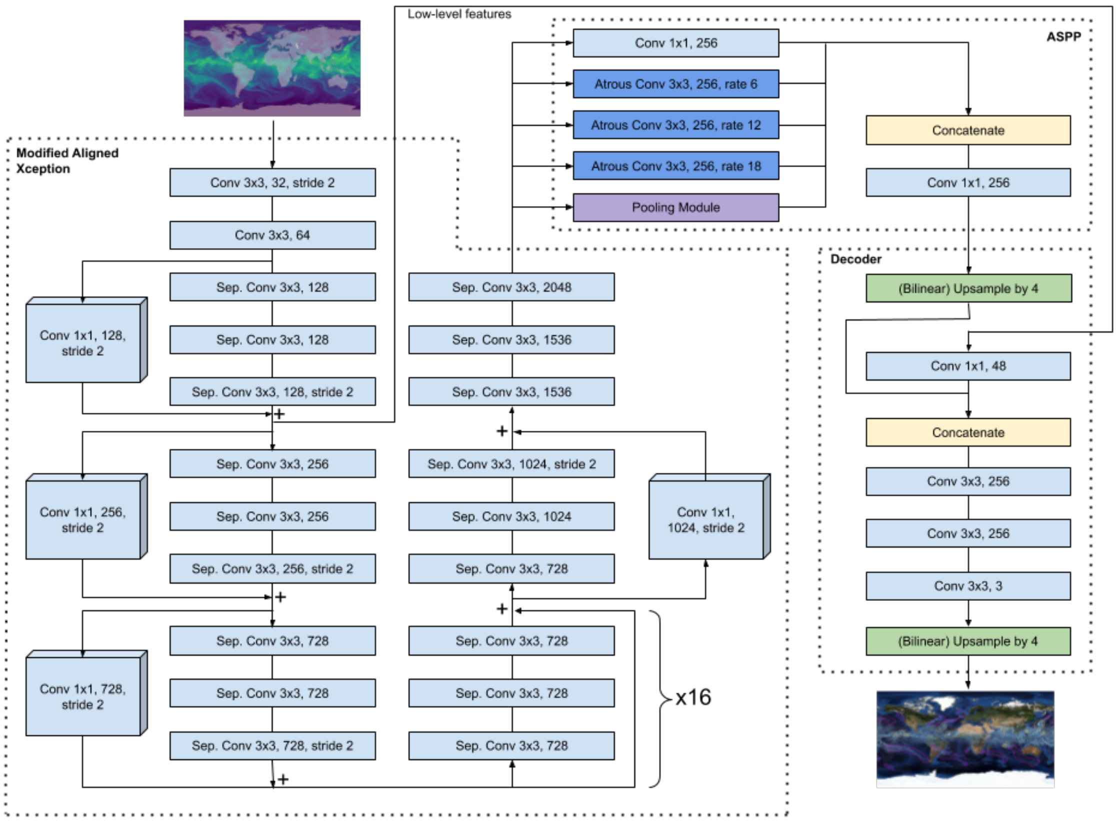


Figure 6.3: DeepLabv3+ network: All convolutional layers are followed by a batch normalization and a ReLU activation layer, which are omitted from this schematic for the sake of brevity. "Sep. Conv" denotes depthwise separable convolution. The pooling module consists of a two-dimensional pooling layer, followed by a convolutional layer, a batch normalization layer and a ReLU activation layer. For more details, we refer the reader to [22].

samples (40% of all samples in $D_1$), and validation and test sets, which are disjoint sets each randomly sampled from the remaining 60%. Dataset $D_2$ is a subset of ClimateNet and consists of 219 unique images, randomly chosen from the same time interval. Each sample in $D_2$ is labeled by *at least* one human expert, that is, there also exist samples which are labeled by multiple experts. The total number of expert-labeled images is 459. The training set for dataset $D_2$ contains 422 (92%) samples; validation and test sets contain 18 (4%) and 19 (4%) samples, respectively.

A unique challenge in applying standard computer vision-based DL architectures to climate problems is that climate images are heavily imbalanced: 94% of the pixels in ClimateNet data correspond to the "background" class. The DL architecture can naively learn a mapping of any input image pixel to the background class, and be correct 94% of the time! In order to account for this unique challenge, we train our network by optimizing the weighted cross-entropy loss function using the Adam optimizer [105]. For such a loss, the class weights are usually defined to be the inverses of class frequencies. However, this choice for the class weights leads to certain numerical issues [113]. In order to circumvent these issues, we use the squared inverses of class frequencies as class weights.

We use a learning rate scheduler that multiplicatively reduces the learning rate each time the performance on the validation set does not improve for 3 epochs in a row, and set the initial learning rate to be $1.5 \times 10^{-3}$. We distribute the training process over 8 GPUs, and use a batch size of 16. For both datasets, we intialize the model with random weights and stop the training as soon as the model's performance on the validation set starts degrading. This corresponds to a training time of 20 epochs for dataset $D_1$, and 5 epochs for dataset $D_2$.

During training, we track the loss incurred by the model on the training and validation sets. We note that while the model did begin to incur larger losses on the validation set, the incurred training loss never converged. From this observation, we conclude that the model has potential to learn the segmentation masks even better, if provided with more hand-labeled data. Comparing the behavior of the training loss curve of our model with those obtained by experiments involving subsets of the dataset $D_1$, we expect the model to begin converging only if the hand-labeled dataset contains at least a few thousand samples.

### 6.2.2 Inference

Once the training phase is over, we obtain two models: model $M_1$ which was trained on dataset $D_1$, and $M_2$ which was trained on $D_2$. We then use these models to run inference on the held-out test sets of $D_1$ and $D_2$, respectively. We also use the model $M_2$ to run inference on a completely different dataset $D_3$, which corresponds to a climate change scenario. We used a single GPU for this process as inference is computationally much more lightweight than training. As described above, the models produce segmentation masks of size $(1152, 768)$ for every sample in their respective test sets. These masks are then used to evaluate the performance of the model. A detailed discussion of the results is reported in Section 6.3.2.

### 6.2.3 Conditional precipitation analyses

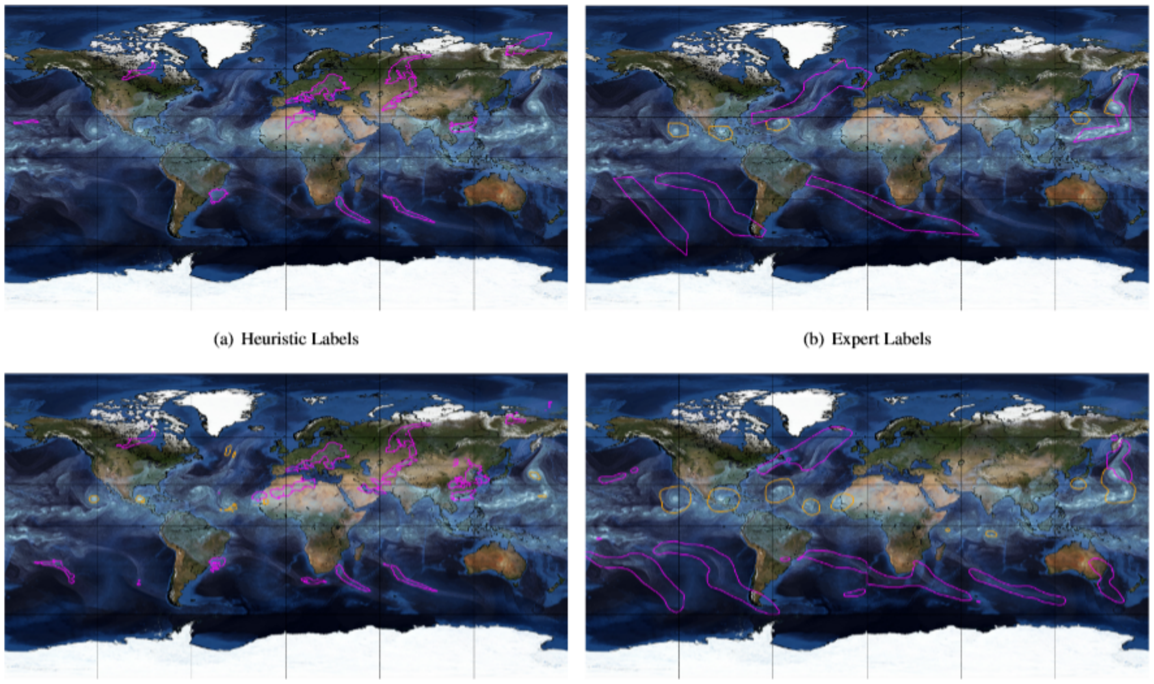
Once Deep Learning has been applied to obtain pixel-level segmentation masks for TCs and ARs, a host of downstream analytics can now be conducted, for example, we can extract and summarize various *conditional* probability distributions associated with individual event types. In this paper we report on global precipitation associated with TCs and ARs and regional precipitation associated with ARs in the state of California, and TCs in the Gulf of Mexico. Further, we present percentiles and scaling relationships due to global warming of extreme precipitation associated with TCs and ARs at global and regional scales.

A key challenge in conducting such highly precise analytics is the requirement to create conditional probability distributions over $O(10M)$-$O(10)B$ pixels, where each pixel contains the value of a physical quantity at a grid point in the climate model output. We leverage the fastKDE package, developed by [167, 166], to compute these distributions efficiently and effectively in seconds to minutes on a single workstation.

## 6.3 Segmentation Results

### 6.3.1 ClimateNet Dataset

The curated expert-labeled ClimateNet dataset, the trained DL segmentation model and PyTorch code to use the model in inference mode are available for download at `https://portal.nersc.gov/project/ClimateNet/`.



(a) Heuristic Labels

(b) Expert Labels

(c) Predictions trained on Heuristic Labels

(d) Predictions trained on Expert Labels

Figure 6.4: Comparison of heuristic labels, expert labels, and DL segmentation model predictions. Heuristic labels from the TECA package [184], as in (a), are used to train a DL segmentation model, $M_1$. Labels from human experts, as in (b), are used to train a DL segmentation model, $M_2$. Segmentation masks predicted from $M_1$ are shown in (c) and those from $M_2$ are shown in (d).

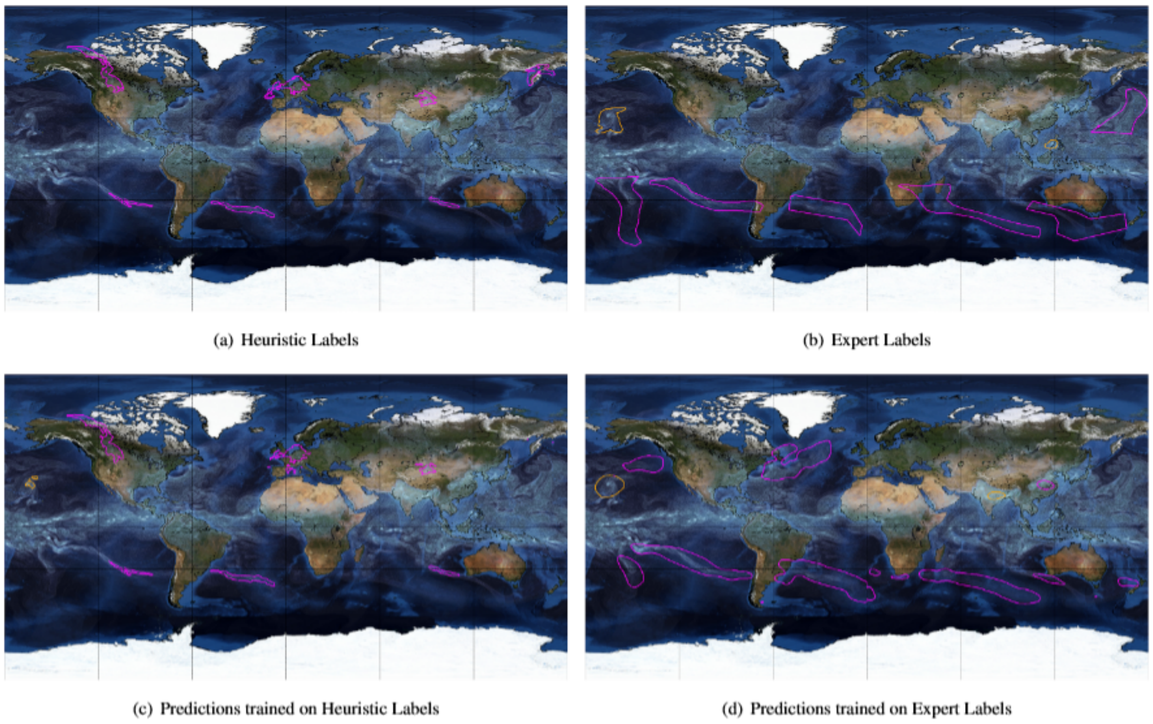

(a) Heuristic Labels
(b) Expert Labels
(c) Predictions trained on Heuristic Labels
(d) Predictions trained on Expert Labels

Figure 6.5: Comparison of heuristic labels, expert labels, and DL segmentation model predictions. Heuristic labels from the TECA package [184], as in (a), are used to train a DL segmentation model, $M_1$. Labels from human experts, as in (b), are used to train a DL segmentation model, $M_2$. Segmentation masks predicted from $M_1$ are shown in (c) and those from $M_2$ are shown in (d).

### 6.3.2 Segmentation Results

The trained DL models are evaluated on two CAM5.1 scenarios - (i) *All-hist* and (ii) the so-called *UNHAPPI* ([265]; [145]; [264]). The *All-hist* scenario runs from 1995-2015 and includes all natural and anthropogenic forcings. The Half a degree Additional warming, Prognosis and Projected Impacts (HAPPI) experimental protocol was designed to compare the effects of stabilizing anthropogenic global warming at 1.5°C and 2.0°C over preindustrial levels(Mitchell et al., 2017). The *UNHAPPI* scenario a stabilizes the anthropogenic warming at 3 °C over preindustrial levels. The details of both scenarios can be found in the listed papers.

#### 6.3.2.1 Qualitative Assessment

We compare visually the performance of the DL models $M_1$ and $M_2$ trained on heuristic labels and human expert labels respectively in Figures 6.4, 6.5 and 6.6. These composite figures

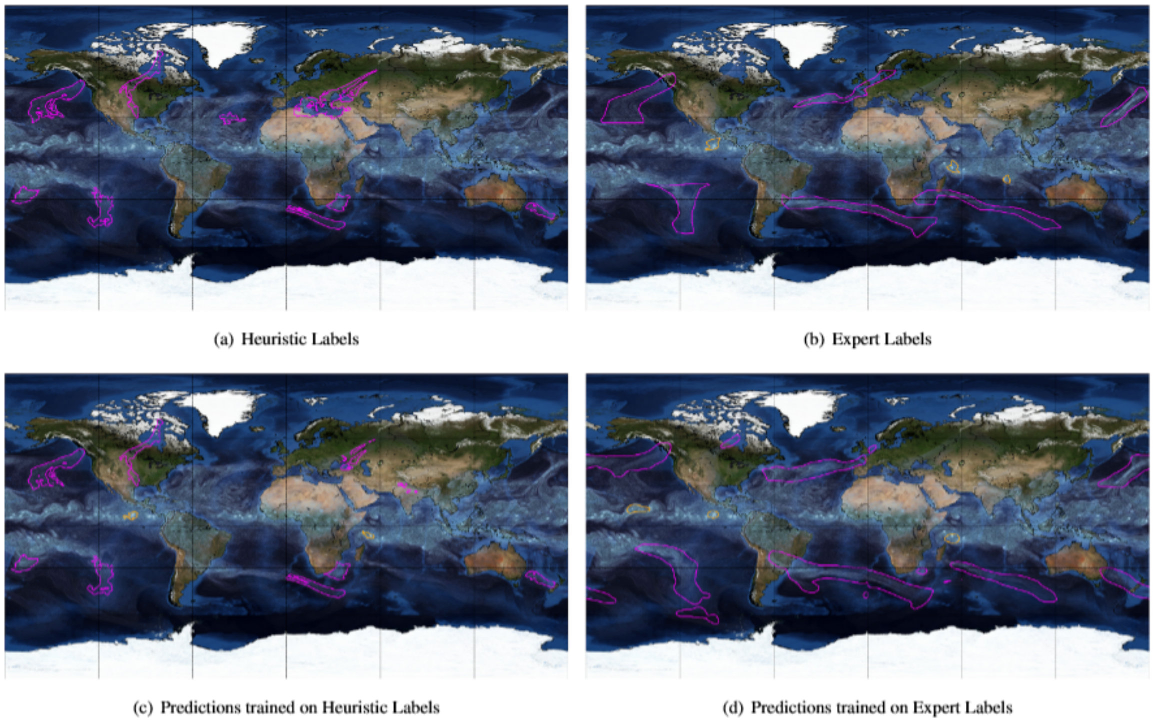

(a) Heuristic Labels
(b) Expert Labels
(c) Predictions trained on Heuristic Labels
(d) Predictions trained on Expert Labels

Figure 6.6: Comparison of heuristic labels, expert labels, and DL segmentation model predictions. Heuristic labels from the TECA package [184], as in (a), are used to train a DL segmentation model, $M_1$. Labels from human experts, as in (b), are used to train a DL segmentation model, $M_2$. Segmentation masks predicted from $M_1$ are shown in (c) and those from $M_2$ are shown in (d).

highlight a few key points: (i) DL models are effective at learning mappings between input images and output pixel masks that exist in the training data, as illustrated by pairs (a,c) and (b,d), *i.e.* they faithfully emulate the data they are trained on; (ii) although the weather and climate communities have thoughtfully developed heuristic algorithms to label TCs and ARs, carefully curated human expert labels seem to be more reliable at capturing both presence or absence of events and their spatial extents; (iii) the DL model trained on human expert labels from ClimateNet, $M_2$, performs better at segmenting TCs and ARs compared to the DL model trained on heuristic labels, $M_1$; and (iv) the DL model $M_2$ predicts high quality segmentation masks for TCs and ARs that are temporally consistent, even though the notion of temporal persistence of TCs and ARs is not incorporated into the training process. Readers are encouraged to further examine rendered movies at: *https://tinyurl.com/unhappi-yt* which showcase the realism and remarkable temporal stability of the segmented TCs and ARs. While there are a few false positives and false negative events, all strong TCs and ARs are successfully detected, segmented and tracked by the DL model $M_2$.

#### 6.3.2.2 Quantitative Assessment

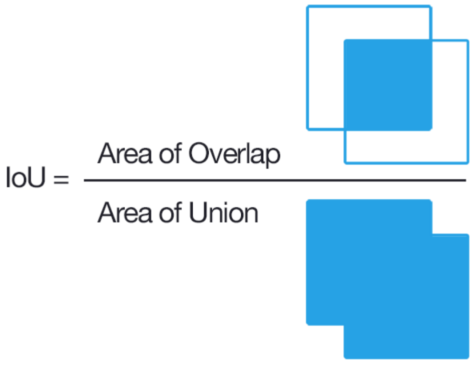


Figure 6.7: Schematic that shows IoU of two square masks. IoU is defined as the ratio of 'the area of the intersection of two segmentation masks' to 'the area of their union'. Note that for the two squares shown here, even an 80% overlap of their edges results in an IoU of 0.47, because the intersection area is 0.64 units and the union is 1.36 units. Source: *https://www.pyimagesearch.com/2016/11/07/intersection-over-union-iou-for-object-detection/*.

We measure the performance of our model using the mean Intersection-over-Union (IoU) metric. Given two binary segmentation masks the IoU is defined as the ratio of 'the area of the intersection of two segmentation masks' to 'the area of their union', as illustrated in Figure. 6.7. While it is a measure of the agreement between two masks, it can be far from unity, especially for masks that are small in size because small disagreements in their overlap are amplified by this measure. Hence we emphasize that IoU not be confused with accuracy. Nevertheless, it is a useful metric for evaluating how well a DL model emulates the characteristics of the data it is trained on. Since we have multiple classes in our study, we calculate a mean IoU metric as the mean of the IoUs of the three classes, *i.e.* AR, TC and background.

Table 6.1 shows comparisons of IoUs obtained for the DL model (DeepLabv3+) trained on labels from heuristics (first row), labels from human experts (second row) and between human experts (third row). In the first and the second row the IoU is calculated on a 'held out' test set that has not been seen by the model during training or validation. For model $M_1$, the IoU is calculated between the model predictions and the heuristic labels, in this case each image has only one set of heuristic labels (from TECA). For model $M_2$, the IoU is calculated between the model predictions and every human expert label that exists for that image (note that the number of human expert labels are not the same for each image), and then averaged. In the third row we calculate, pair-wise, the IoU between every pair of human expert labels for a given image, and then average over all images. The third row gives a measure of how well any two experts agree on their labels, and we use this as the

| IoU Comparisons | | | | |
|---|---|---|---|---|
| | mean IoU | Background IoU | TC IoU | AR IoU |
| Model trained on heuristics, $M_1$ | 0.7354 | 0.9958 | 0.4438 | 0.7667 |
| Model trained on ClimateNet, $M_2$ | 0.5247 | 0.9389 | 0.2441 | 0.3910 |
| Mean IoU between human experts | 0.5120 | 0.9382 | 0.2567 | 0.3412 |

Table 6.1: Model $M_1$ achieves IoU scores similar to those reported in [113]. The overall mean IoU is limited by the TC IoU. After training on ClimateNet, model $M_2$ performs similar to human experts.

target metric that the DL model $M_2$ aims to achieve, *i.e.* we train the model to perform similarly to a human expert.

Given clear, deterministic ground truth labels, where the exact boundaries of every event of each class are well-defined and not subject to discrepancies or uncertainties, the mean IoU is a useful quantitative metric for assessing the quality of segmentation techniques. In our context, however, because the boundaries of ARs and TCs can be hard to define exactly with certainty, it is useful to compare human experts against each other to obtain a measure of the mean IoU between any two human experts, before evaluating performance of the Deep Learning model against human experts.

In Figure 6.8 we show an example of the comparison between two human experts for one snapshot. The background class is most dominant because TCs and ARs occupy a small fraction of the total number of pixels on any given image, hence IoUs for background tend to be quite high, as seen in Table 6.1. However, for TCs and ARs, the IoU can drop to significantly lower values because minor differences in event boundaries for small events can result in low IoU values, as illustrated in Figure 6.8. Even though the experts appear to agree reasonably on their event labels and masks, the mean IoU for these two human experts is 0.59. These results are comparable to those reported in [113].

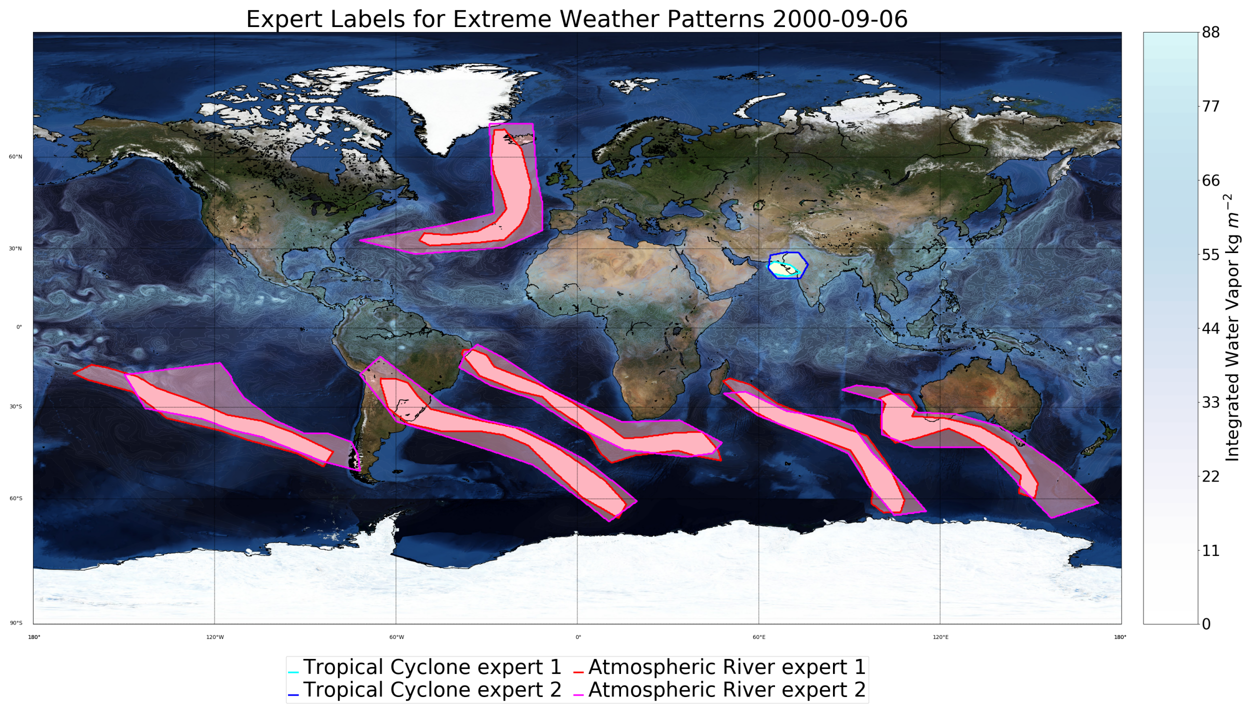

Expert Labels for Extreme Weather Patterns 2000-09-06
60°N
30°N
0°
30°S
60°S
90°S
180°
120°W
60°W
0°
60°E
120°E
180°
88
77
66
55
44
33
22
11
0
Integrated Water Vapor kg $m^{-2}$
Tropical Cyclone expert 1
Tropical Cyclone expert 2
Atmospheric River expert 1
Atmospheric River expert 2

Figure 6.8: Comparison of two different expert labelings with an IoU of 0.59. Note that even though, visually, the experts appear to agree to a large extent on their labels of AR and TC events in this snapshot, the quantitative IoU metric is only 0.59. Hence we emphasize that IoU not be confused with accuracy; good agreement even amongst expert labelers can result in IoU values far from unity. A perfect match, *i.e.* IoU = 1 only results when two labelers agree on every single pixel of the image. As is apparent from this figure, even relatively minor differences in the labels for TCs can disproportionately impact the mean IoU.

# Chapter 7

# Conclusions and Future Work

## 7.1 Methods for Analyzing Extremes

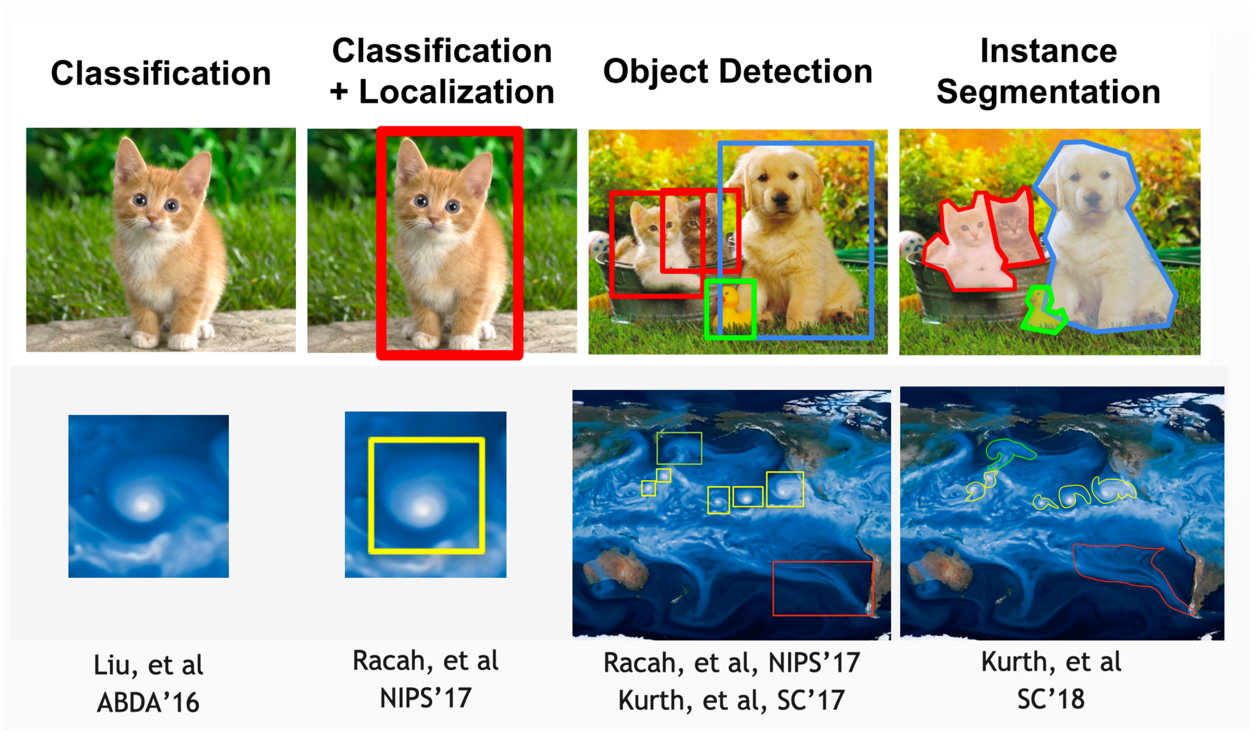


Figure 7.1: Progress made in this thesis in solving a range of analytics problems with Deep Learning

Since the 1980s, the climate science community has used heuristics for defining extreme weather patterns. Despite extensive inter-comparison efforts (ARTMIP, IMILAST, etc) the path forward (i.e. development of yet more heuristics, optimal combination of output of existing heuristics) has been unclear. This thesis has conclusively demonstrated that modern Deep Learning methods are very much relevant for Climate Science, and effective at solving a range of analytics problems (highlighted in Figure 7.1). Through a series of influential publications over the course of this thesis, we have developed state-of-the-art methods for extreme weather identification (classification, detection, segmentation). We would like to encourage other researchers to apply these methods to a range of other extreme weather phenomena (blocking events, mesoscale convective complexes, weather fronts, tornadoes, etc), and build capabilities for *tracking* events in datasets. An open challenge is performing all analyses natively in 3D, which will likely be a computationally daunting task.

We have not been able to develop and apply methods for visualization, interpretability and explainability of trained DL networks. We acknowledge that the proposition of replacing imperfect, hand-tuned heuristics with a near-perfect black-box may not appeal to everyone in the community. We have reason to believe that techniques for visualizing and explaining the behavior of DL networks will be developed quickly by the Machine Learning research community in the near future.

## 7.2 Scaling Methods to Large Datasets

### 7.2.1 TECA

Through the design and execution of the TECA framework, we believe that we have fundamentally enabled the climate science community to apply 'conventional' heuristics to contemporary datasets from CMIP-5 and CMIP-6. While we have developed and applied heuristics for detecting Tropical Cyclones, Atmospheric Rivers and Extra-Tropical Cyclones in this thesis, other researchers in the community have leveraged the framework to develop detectors for jet streams, blocking events, and probabilistic segmentation capabilities. Needless to say, the framework could be applied to other heuristics corresponding to known extreme weather events, and other patterns in the earth sciences (ocean eddies, crop vegetation, land use, etc) broadly.

TECA embodies modern concepts from the 'Big Data' revolution in computer science, and enables the climate informatics community to truly leverage the power of HPC resources. We have conclusively demonstrated the capability of TECA to scale to the largest CPU-based HPC systems in the DOE complex. We believe that more work will be required to optimize the framework for efficient execution on heterogeneous systems (comprising of CPUs, GPUs and accelerators). The primary bottlenecks at this stage is no longer the analytics, but rather downloading geographically disperse datasets to a single system. For instance, while we were able to analyze the entire CMIP-5 archive for extra-tropical cyclones in 1.5 hours on 750K Mira cores, it took us 3 months to download the dataset from various ESGF endpoints. Therefore in the long run, it is clear that supporting analyses capabilities co-hosted with storage systems (e.g. ESGF infrastructure, or NASA/DOE web portals) may be required to ensure broader applicability to the climate science informatice community.

### 7.2.2 Deep Learning on CPUs

This thesis has presented the first 15-PetaFLOP Deep Learning software running on HPC platforms. We have utilized IntelCaffe to obtain ∼2 TF on single Xeon Phi nodes. We utilize a hyrid strategy employing synchronous groups and asynchronous communication among them to scale the training of a single model to 9600 Cori Phase II nodes. We apply this framework to solve real-world supervised and semi-supervised patterns classification problems in HEP and Climate Science. Our work demonstrates that manycore HPC platforms can be successfully used to accelerate Deep Learning, opening the gateway for broader adoption by the domain science community.

#### 7.2.2.1 Deep Learning on HPC

To the best of our knowledge, our work is the first successful attempt at scaling Deep Learning on CPU-based HPC systems. We share a number of insights from this unique exercise.

First, at a scale of thousands of nodes, we found significant variability in runtimes across runs, which could be as high as 30%. The probability of one of the thousands of nodes failing or degrading during the run is non-zero. In this work, we report runs where we did not encounter complete node failures. We note that even a single node failure can cause complete failure of synchronous runs; hybrid runs are much more resilient since only one of the compute groups gets affected. However, even in hybrid runs, if model updates from one of the compute groups lags significantly behind others, it can result in "jumps" in the overall loss and accuracy.

Second, current architectures and software stacks for deep learning are still not as mature as the traditional HPC application stack. Specifically, performance on small batch sizes (essential for scale out) has not been completely optimized in many frameworks. Further, the state of the art in deep learning kernel implementations is rapidly evolving with new algorithms like Winograd [123] and FFT based algorithms. We did not experiment with such algorithms in this work; studying the impact on per-node performance and scale-out behaviour of these algorithms is a direction for future research.

There has been a lot of discussion surrounding training with quantized weights and activations [82, 32]. The statistical implications of low precision training are still being explored [66, 67], with various forms of stochastic rounding being of critical importance in convergence. While supercomputers with architectures supporting low precision computations in hardware are not yet present, we believe that such systems have the potential to further accelerate training time for our applications.

#### 7.2.2.2 Deep Learning for Science

We believe that science domains that can readily generate vast amounts of representative training data (via simulators) stand to benefit immediately from progress in DL methods. In other scientific domains, unsupervised and semi-supervised learning are key challenges for the future. In both cases, it is unreasonable to expect scientists to be conversant in the art of hyper-parameter tuning. Hybrid schemes add to the complexity of the tuning process. It is therefore critical that, in addition to optimized libraries such as IntelCaffe, MKL/MKL-DNN and MLSL, higher-level libraries such as Spearmint [228] be provided for automating the search for network architectures. We also note that more aggressive optimizations involving computing in low-precision and communicating high-order bits of weight updates are poorly understood with regards to their implications for classification and regression accuracy for *scientific* datasets. A similar story holds with regards to deployment of DL models. Unlike commercial applications where a sparse/compact representation of the model needs to be deployed in-situ, scientific applications will typically utilize DL models within the context of the HPC/Datacenter environment.

### 7.2.3 Exascale Deep Learning

We have presented the first exascale-class deep learning application in the world. Motivated by the important problem of segmenting extreme weather patterns, we have successfully applied the Tiramisu and DeepLabv3+ architectures to high resolution, multi-variate climate datasets. We developed a number of enhancements to the deep learning algorithms (custom loss function, optimization schemes, channels and network architecture) to obtain excellent qualitative and quantitative results. We built upon a number of system-level optimizations (advanced data staging, optimized data ingestion, and hierarchical all-reduce communication) to scale the scientific application to an unprecedented level of concurrency (4560 Summit nodes, 27,360 Volta GPUs), scaling efficiency (90%) and performance (1.13 EF/s peak, 999, PF/s sustained). Our work extends open-source TensorFlow and Horovod tools, thereby benefiting the broader scientific and commercial deep learning communities. The environment we have developed is already in use by other teams on Summit and the methodologies will extend to current and future HPC platforms.

#### 7.2.3.1 Implications for Future Systems

Scaling Deep Learning further on future exascale machines with purely data parallel techniques will prove to be numerically difficult. Techniques such as LARC have increased the total global batch size that can converge, but we view the incorporation of model parallel approaches as being indispensable in the foreseeable future. Systems like Summit (with high speed NVLink connections between processors) are amenable to domain decomposition techniques that split layers across processors. Exploring model parallel implementation across nodes is a natural extension that will require investments in more complex collectives in software libraries, like Horovod and NCCL, and optimizations at the algorithm as well at network switch level. Additional training performance optimizations will increase the rate at which we need to feed input data to the networks, further exacerbating the parallel I/O problem. While compression techniques can be used at the expense of already heavily utilized main processors, more memory close to the compute elements, such as node-local non-volatile memory, may help reduce the pressure on the global file system. There is also a potential for processing at the storage layer itself to aid in data processing and augmentation. Generally the stress on the dataplane and communication layers will quickly increase, requiring holistic approaches towards hardware and software co-design.

To conclude, we believe that field of Deep Learning is poised to have a major impact on the scientific world. Scientific data requires training and inference at scale, and while Deep Learning might appear to be a natural fit for existing petascale and future exascale HPC systems, careful consideration must be given towards balancing various subsystems (CPUs, GPUs/accelerators, memory, storage, I/O and network) to obtain high scaling efficiencies. Sustained investments are required in the software ecosystem to seamlessly utilize algorithmic innovations in this exciting, dynamic area.

## 7.3 ClimateNet

This thesis has demonstrated conclusively that Deep Learning models trained on curated expert-labeled climate data – ClimateNet – are powerful tools for segmenting extreme weather patterns in climate datasets, enabling precision climate data analytics. We have developed an end-to-end infrastructure for acquiring expert-labeled data (via ClimateContours); curating the data carefully (using rigorous QA+QC protocols); training DL segmentation models; running DL segmentation models in inference mode; and conducting downstream conditional precipitation analyses.

The proposed dataset – ClimateNet – and end-to-end infrastructure provides several unique capabilities: (i) it enables us to perform fine-grained highly precise data analytics, such as examining changes in frequency and intensity of weather patterns at specific geographic locations across the globe; (ii) it can be applied to different climate scenarios and different datasets without tuning since it does not rely on threshold conditions unlike heuristic algorithms currently used in the community; (iii) the method is suitable for rapidly analyzing large amounts of climate model output. Further, the method can likely be used directly with reanalyses products or observational data using transfer learning, as shown successfully for a similar DL-based method by [70]. While we do not explicitly test the transferability of this model to observations and reanalyses products, we intend to pursue this in future work.

Our work highlights the advantages of transitioning to modern, data-driven DL methods for hi-precision climate data analytics. While our results are promising, we highlight current limitations in our methodology and identify opportunities for future studies:

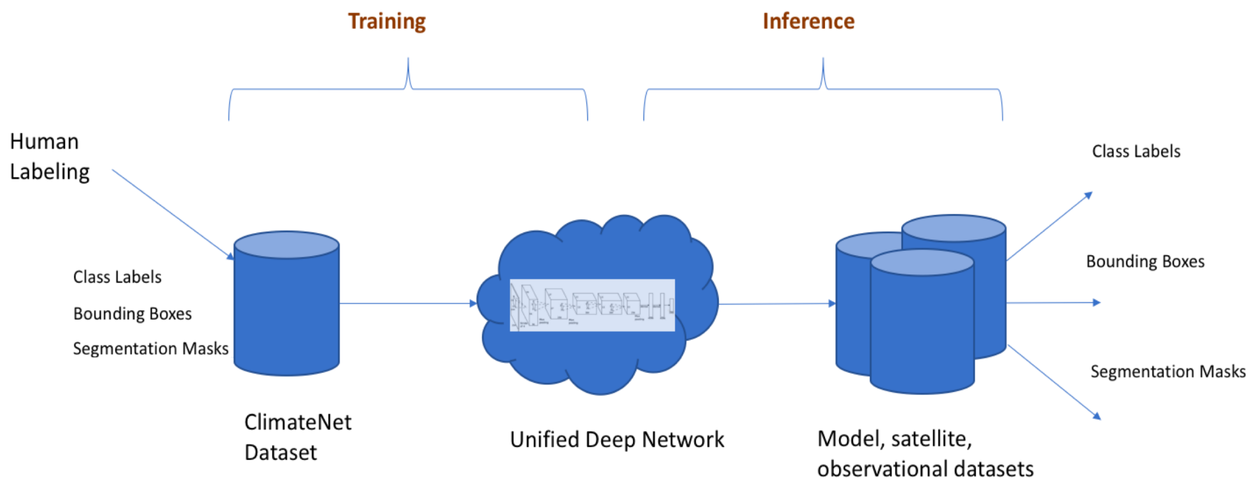


Figure 7.2: Overall vision for applying ClimateNet dataset and architecture for the broader climate science community

1. *Limited Training Data:* The quality of our segmentation results is fundamentally limited by access to large amounts of expert-labeled data. We have only been able to curate ≈600 expert-labelled images thus far, and while the resulting DL model does a reasonable job on held-out datasets, we expect that the performance will be improved further with larger amounts of curated expert-labelled data. We appeal to the climate science community to contribute labels to the ClimateNet project – an open source, community project – which is live and freely usable by anyone world-wide at `https://www.nersc.gov/research-and-development/data-analytics/big-data-center/climatenet/`.

   - *Applicability of Transfer Learning:* We intended to leverage the relatively large amount of training data available via AR and TC pattern detection heuristics, such as TECA [182], and a smaller amount of ClimateNet labeled data via Transfer Learning [277] to train a model first on heuristics-based training samples followed by "fine-tuning" using ClimateNet data. This approach, however, produced a model that was less skillful at segmenting ARs and TCs compared to a model trained purely on ClimateNet data and further work is needed to understand whether alternative transfer learning techniques are required to obtain more accurate results.

   - *Applicability of Curriculum Learning*: It has been shown that curriculum learning ([132, 267]), a type of learning process where a DL model learns to perform well on simpler tasks first before progressively learning harder tasks, is an effective approach for learning complex tasks with limited data. We intend to employ such techniques, for example, by training on cropped centered snapshots of single events before learning on fully global hi-resolution datasets, and design curricula for efficient and effective learning with limited data.

- *Applicability of Active Learning to prioritize images for labeling:* Our current procedure for choosing candidate climate datapoints for experts to label is unweighted and at random. In particular, we do not choose 'easy' vs. 'hard' images, nor does labeling $N$ images inform the choice of the $(N+1)^{th}$ image presented to a human expert for labeling. In the future, we intend to explore adaptive strategies for downselecting and prioritizing images for manually intensive labeling.

2. *Spatio-Temporal Segmentation:* Our current segmentation models are purely spatial in nature, and do not take temporal persistence of weather events into account. It is, indeed, quite remarkable how well these purely spatial models perform in tracking coherent features through time. To minimize false positives and false negatives and capture more faithfully the evolution of these coherent structures, we intend to augment DL models with consecutive snapshots. There are, however, implications for acquiring expert labels for many more datapoints, and accommodating and training large DL models on GPUs that may require data parallelism and/or model parallelism.

3. *Assessing performance on other types of Climate Data:* Thus far, we have only trained and tested our DL models on CAM5.1, 25-km resolution data. We intend to systematically explore whether the trained model can be applied to: (i) CAM5 output at different spatial and temporal resolutions; (ii) other weather and climate models at comparable resolutions; and (iii) observational and reanalyses products. Given that Deep Learning models learn complex feature representations at multiple levels of abstraction, they will likely work well across modalities, but this generalization claim needs to be tested explicitly.

4. *Probabilistic Segmentation:* We currently acquire labels for every climate snapshot from many human experts with self-ratings on their level of confidence (high, medium, or low) for every event (TC or AR). We do not, however, incorporate these self-ratings into the training procedure, for example, as a form of uncertainty. Building on the work of [136], we intend to use the multiple expert labels weighted suitably by their self-ratings for every event to predict pixel-wise probabilistic segmentation masks.

5. *Hypothesis Testing:* Thus far, we have presented preliminary results on changes in extreme precipitation, and associated CC-scaling relationships. One of the unique capabilities provided by our framework is the possibility of rapidly exploring hypotheses related to dynamical mechanisms (the concept is illustrated in Figure 7.3). For instance, we can index into dynamical variables such as moisture convergence on a per-storm basis; correlate that information with precipitation; and test hypotheses regarding local dynamical mechanisms being responsible for super-CC scaling. More advanced versions of hypothesis testing could relate dynamical interactions between jet-streams, extra-tropical cyclones and atmospheric rivers.

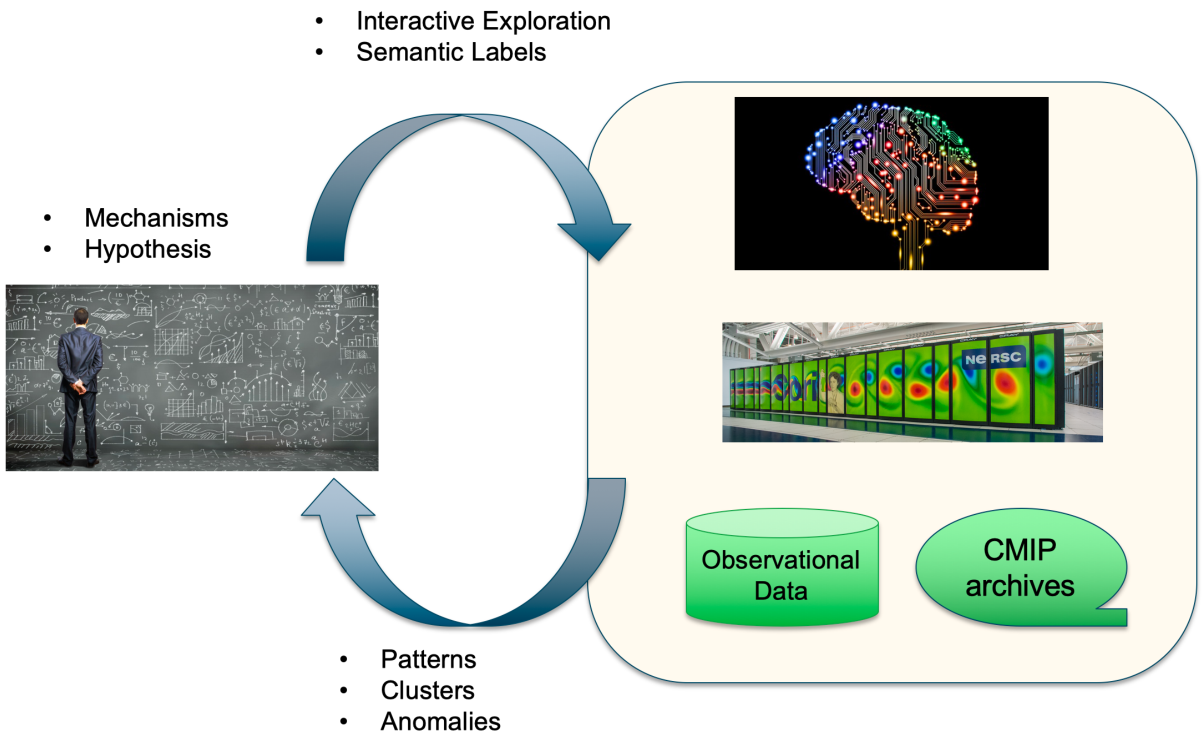


Figure 7.3: Scalable AI techniques will make the process of hypothesis exploration more seamless in the future. Towards accomplishing this goal, this thesis has successfully bridged the gap between state-of-the-art in computer science and climate science.

# Bibliography

[1] Takuya Akiba, Keisuke Fukuda, and Shuji Suzuki. "ChainerMN: Scalable Distributed Deep Learning Framework". In: *Proceedings of Workshop on ML Systems in The Thirty-first Annual Conference on Neural Information Processing Systems (NIPS)*. 2017. URL: http://learningsys.org/nips17/assets/papers/paper_25.pdf.

[2] Takuya Akiba, Shuji Suzuki, and Keisuke Fukuda. "Extremely Large Minibatch SGD: Training ResNet-50 on ImageNet in 15 Minutes". In: (2017). URL: https://www.preferred-networks.jp/docs/imagenet_in_15min.pdf.

[3] Myles Allen and William Ingram. "Constraints on Future Changes in Climate and the Hydrologic Cycle". In: *Nature* 419 (Oct. 2002), pp. 224–32. DOI: 10.1038/nature01092.

[4] Dario Amodei et al. "Deep Speech 2 : End-to-End Speech Recognition in English and Mandarin". In: *Proceedings of ICML)*. 2016, pp. 173–182.

[5] Anima Anandkumar. *Deep Learning at Scale on AWS*. URL: https://ml-days-prd.s3.amazonaws.com/slides/speakers/slides/3/Anima-EPFL2017.pdf (visited on 04/13/2017).

[6] Awni Y. Hannun Andrew L. Maas and Andrew Y. Ng. "Rectifier Nonlinearities Improve Neural Network Acoustic Models". In: *ICML Workshop on Deep Learning for Audio, Speech, and Language Processing* (2013).

[7] Nicolas Ballas et al. "Delving Deeper into Convolutional Networks for Learning Video Representations". In: *In the Proceedings of ICLR. arXiv preprint arXiv:1511.06432* (2016).

[8] L. Bengtsson, K. I. Hodges, and N. Keenlyside. "Will extra-tropical storms intensify in a warmer climate?" In: *J. Climate* 22 (2009), pp. 2276–2301.

[9] L. Bengtsson, K. I. Hodges, and E. Roeckner. "Storm tracks and climate change". In: *J. Climate* 19 (2006), pp. 3518–3543.

[10] Elsa Bernard et al. "Clustering of Maxima: Spatial Dependencies among Heavy Rainfall in France". In: *J. Climate* 26 (2013), pp. 7929–7937.

[11] Noah D Brenowitz and Christopher S Bretherton. "Prognostic validation of a neural network unified physics parameterization". In: *Geophysical Research Letters* 45.12 (2018), pp. 6289–6298.

[12] Eric Brochu, Vlad M Cora, and Nando De Freitas. "A tutorial on Bayesian optimization of expensive cost functions, with application to active user modeling and hierarchical reinforcement learning". In: *arXiv preprint arXiv:1012.2599* (2010).

[13] Surendra Byna et al. "Detecting Atmospheric Rivers in Large Climate Datasets". In: *Proceedings of the 2Nd International Workshop on Petascal Data Analytics: Challenges and Opportunities*. PDAC '11. Seattle, Washington, USA: ACM, 2011, pp. 7–14. ISBN: 978-1-4503-1130-4. DOI: 10.1145/2110205.2110208. URL: http://doi.acm.org/10.1145/2110205.2110208.

[14] *CDAT*. http://www-pcmdi.llnl.gov/software-portal/cdat.

[15] Edmund Chang. "CMIP5 Projection of Significant Reduction in Extratropical Cyclone Activity over North America". In: *Journal of Climate* 26 (Dec. 2013), pp. 9903–9922. DOI: 10.1175/JCLI-D-13-00209.1.

[16] Fu Chang, Chun-Jen Chen, and Chi-Jen Lu. "A linear-time component-labeling algorithm using contour tracing technique". In: *Comput. Vis. Image Underst.* 93.2 (2004), pp. 206–220.

[17] WE Chapman et al. "Improving Atmospheric River Forecasts With Machine Learning". In: *Geophysical Research Letters* 46.17-18 (2019), pp. 10627–10635.

[18] Rajib Chattopadhyay, Augustin Vintzileos, and Chidong Zhang. "A description of the Madden–Julian Oscillation based on a self-organizing map". In: *Journal of Climate* 26.5 (2013), pp. 1716–1732.

[19] Daniel Chavas, Ning Lin, and Kerry Emanuel. "A Model for the Complete Radial Structure of the Tropical Cyclone Wind Field. Part I: Comparison with Observed Structure*". In: *Journal of the Atmospheric Sciences* 72 (July 2015). DOI: 10.1175/JAS-D-15-0014.1.

[20] Chenyi Chen et al. "Deepdriving: Learning affordance for direct perception in autonomous driving". In: *Proceedings of the IEEE International Conference on Computer Vision*. 2015, pp. 2722–2730.

[21] Liang-Chieh Chen et al. "Encoder-Decoder with Atrous Separable Convolution for Semantic Image Segmentation". In: *ECCV*. 2018.

[22] Liang-Chieh Chen et al. "Encoder-Decoder with Atrous Separable Convolution for Semantic Image Segmentation". In: *arXiv e-prints*, arXiv:1802.02611 (Feb. 2018), arXiv:1802.02611. arXiv: 1802.02611 [cs.CV].

[23] Trishul Chilimbi et al. "Project adam: Building an efficient and scalable deep learning training system". In: *11th USENIX Symposium on Operating Systems Design and Implementation (OSDI 14)*. 2014, pp. 571–582.

[24] François Chollet. "Xception: Deep Learning with Depthwise Separable Convolutions". In: *arXiv e-prints*, arXiv:1610.02357 (Oct. 2016), arXiv:1610.02357. arXiv: 1610.02357 [cs.CV].

[25] *CliVAR Hurricane Group*. http://www.usclivar.org/working-groups/hurricane.

[26] C. A. S. Coelho et al. "Methods for Exploring Spatial and Temporal Variability of Extreme Events in Climate Data". In: *J. Climate* 21.10 (2008), pp. 2072–2092.

[27] S. G. Coles. *An Introduction to Statistical Modeling of Extreme Values*. New York: Springer Verlag, 2001.

[28] *Community Earth System Model*. `http://www.cesm.ucar.edu/working-groups/Atmosphere/development`.

[29] Andrew J Conley et al. "Description of the NCAR community atmosphere model (CAM 5.0)". In: (2012).

[30] D. Cooley, P. Naveau, and P. Poncet. "Variograms for spatial max-stable random fields". In: vol. 187. Lecture Notes in Statistics. Springer, 2006, pp. 373–390.

[31] *Coupled Model Intercomparison Project Phase 5*. http://cmip-pcmdi.llnl.gov/cmip5/.

[32] Matthieu Courbariaux, Yoshua Bengio, and Jean-Pierre David. "Training deep neural networks with low precision multiplications". In: *CoRR* abs/1412.7024 (2014).

[33] George E Dahl et al. "Context-dependent pre-trained deep neural networks for large-vocabulary speech recognition". In: *Audio, Speech, and Language Processing, IEEE Transactions on* 20.1 (2012), pp. 30–42.

[34] Dipankar Das et al. "Distributed Deep Learning Using Synchronous Stochastic Gradient Descent". In: *CoRR* abs/1602.06709 (2016).

[35] Jeffrey Dean et al. "Large scale distributed deep networks". In: *NIPS*. 2012, pp. 1223–1231.

[36] *DeepBench*. `github.com/baidu-research/DeepBench`. 2017.

[37] M. D. Dettinger. "Climate Change, Atmospheric Rivers, and Floods in California – A Multimodel Analysis of Storm Frequency and Magnitude Change". In: *Journal of the American Water Resources Association* 47.3 (2011), pp. 514–523.

[38] M. D. Dettinger. *Fifty-two years of "pineapple-express" storms across the West Coast of North America*. Tech. rep. CEC-500-2005-004, 20pp. U.S. Geological Survey, Scripps Institution of Oceanography for the California Energy Commission, PIER Project Rep, 2004.

[39] Michael D Dettinger et al. "Atmospheric rivers, floods and the water resources of California". In: *Water* 3.2 (2011), pp. 445–478.

[40] Michael D Dettinger et al. "Atmospheric rivers, floods and the water resources of California". In: *Water* 3.2 (2011), pp. 445–478.

[41] Michael B. Dillencourt, Hannan Samet, and Markku Tamminen. "A general approach to connected-component labeling for arbitrary image representations". In: *J. ACM* 39.2 (1992), pp. 253–280.

[42] Jeffrey Donahue et al. "Long-Term Recurrent Convolutional Networks for Visual Recognition and Description". In: *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2015.

[43] John Dutton. "Opportunities and Priorities in a New Era for Weather and Climate Services". In: *Bulletin of the American Meteorological Society* 83 (Sept. 2002). DOI: 10.1175/1520-0477(2002)083<1303:OAPIAN>2.3.CO;2.

[44] *Earth System Grid Federation*. http://pcmdi9.llnl.gov/esgf-web-fe/.

[45] Samira Ebrahimi Kahou et al. "Recurrent neural networks for emotion recognition in video". In: *Proceedings of the 2015 ACM on International Conference on Multimodal Interaction*. ACM. 2015, pp. 467–474.

[46] Vicky Espinoza et al. "Global analysis of climate change projection effects on atmospheric rivers". In: *Geophysical Research Letters* 45.9 (2018), pp. 4299–4308.

[47] *Ferret Data Visualization and Analysis*. http://ferret.wrc.noaa.gov/Ferret/.

[48] Joel Finnis et al. "Response of Northern Hemisphere extratropical cyclone activity and associated precipitation to climate change, as represented by the Community Climate System Model". In: *Journal of Geophysical Research* 112 (Nov. 2007). DOI: 10.1029/2006JG000286.

[49] Holger Flatt et al. "A parallel hardware architecture for connected component labeling based on fast label merging". In: *ASAP*. IEEE Computer Society, 2008, pp. 144–149. URL: http://dx.doi.org/10.1109/ASAP.2008.4580169.

[50] Eva M. Furrer et al. "Statistical modeling of hot spells and heat waves". In: *Climate Research* 43 (2010), pp. 191–205.

[51] Yang Gao et al. "Dynamical and thermodynamical modulations on future changes of landfalling atmospheric rivers over western North America". In: *Geophysical Research Letters* 42.17 (2015), pp. 7179–7186.

[52] Daniel George and E. A. Huerta. "Deep neural networks to enable real-time multimessenger astrophysics". In: *Phys. Rev. D* 97 (4 Feb. 2018), p. 044039. DOI: 10.1103/PhysRevD.97.044039.

[53] Alexander Gershunov et al. "Precipitation regime change in Western North America: the role of Atmospheric Rivers". In: *Scientific reports* 9.1 (2019), pp. 1–11.

[54] B. Ginsburg, I. Gitman, and O. Kuchaiev. "Layer-Wise Adaptive Rate Control for Training of Deep Networks". In: *in preparation* (2018).

[55] Ross Girshick et al. "Rich feature hierarchies for accurate object detection and semantic segmentation". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2014, pp. 580–587.

[56] Xavier Glorot and Yoshua Bengio. "Understanding the difficulty of training deep feedforward neural networks". In: *International conference on artificial intelligence and statistics*. 2010, pp. 249–256.

[57] Rafael Gómez-Bombarelli et al. “Automatic chemical design using a data-driven continuous representation of molecules”. In: *CoRR* abs/1610.02415 (2016). arXiv: 1610.02415. URL: http://arxiv.org/abs/1610.02415.

[58] Rafael C. Gonzalez and Richard E. Woods. *Digital Image Processing*. 2nd. New Jerse: Prentice Hall, 2002.

[59] Ian Goodfellow et al. “Generative adversarial nets”. In: *Advances in Neural Information Processing Systems*. 2014, pp. 2672–2680.

[60] Jorge Gorricha, Victor Lobo, and Ana Cristina Costa. “A framework for exploratory analysis of extreme weather events using geostatistical procedures and 3D self organizing maps”. In: *International Journal on Advances in Intelligent Systems* 6.1 (2013).

[61] Raghav Goyal et al. “The "something something" video database for learning and evaluating visual common sense”. In: *arXiv preprint arXiv:1706.04261* (2017).

[62] Alan Graves, Abdel-rahman Mohamed, and Geoffrey Hinton. “Speech recognition with deep recurrent neural networks”. In: *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*. IEEE. 2013, pp. 6645–6649.

[63] John Greiner. “A comparison of parallel algorithms for connected components”. In: *SPAA '94: Proceedings of the sixth annual ACM symposium on Parallel algorithms and architectures*. Cape May, New Jersey, United States: ACM, 1994, pp. 16–25.

[64] Richard Grotjahn and Ghislain Faure. “Composite Predictor Maps of Extraordinary Weather Events in the Sacramento California Region”. In: *Weather and Forecasting* 23.3 (2008), pp. 313–335.

[65] Chunhui Gu et al. “AVA: A Video Dataset of Spatio-temporally Localized Atomic Visual Actions”. In: *arXiv preprint arXiv:1705.08421* (2017).

[66] Suyog Gupta et al. “Deep Learning with Limited Numerical Precision”. In: *CoRR* abs/1502.02551 (2015).

[67] Philipp Gysel, Mohammad Motamedi, and Soheil Ghiasi. “Hardware-oriented Approximation of Convolutional Neural Networks”. In: *CoRR* abs/1604.03168 (2016).

[68] L. de Haan. “A spectral representation for max-stable processes”. In: *Annals of Probability* 12.4 (1984), pp. 1194–1204.

[69] Stefan Hadjis et al. “Omnivore: An optimizer for multi-device deep learning on cpus and gpus”. In: *arXiv:1606.04487* (2016).

[70] Yoo-Geun Ham, Jeong-Hwan Kim, and Jing-Jia Luo. “Deep learning for multi-year ENSO forecasts”. In: *Nature* 573.7775 (2019), pp. 568–572.

[71] B. Harvey, Len Shaffrey, and T. Woollings. “Equator-to-pole temperature differences and the extra-tropical storm track responses of the CMIP5 climate models”. In: *Climate Dynamics* 43 (July 2013). DOI: 10.1007/s00382-013-1883-9.

[72] Daren Hasenkamp et al. “Finding Tropical Cyclones on a Cloud Computing Cluster: Using Parallel Virtualization for Large-Scale Climate Simulation Analysis”. In: *CloudCom*. 2010.

[73] K.A. Hawick, A. Leist, and D.P. Playne. “Parallel graph component labelling with GPUs and CUDA”. In: *Parallel Computing* 36.12 (2010), pp. 655–678. ISSN: 0167-8191. DOI: 10.1016/j.parco.2010.07.002. URL: http://www.sciencedirect.com/science/article/B6V12-50RP1TP-1/2/7e10df733f7fc89cf803aab74f8aec84.

[74] Kaiming He et al. “Deep residual learning for image recognition”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 770–778.

[75] Kaiming He et al. “Delving deep into rectifiers: Surpassing human-level performance on imagenet classification”. In: *Proceedings of the IEEE International Conference on Computer Vision*. 2015, pp. 1026–1034.

[76] Alexander Heinecke et al. “LIBXSMM: Accelerating Small Matrix Multiplications by Runtime Code Generation”. In: *Proceedings of SC16*. Salt Lake City, Utah: IEEE Press, 2016, 84:1–84:11. ISBN: 978-1-4673-8815-3.

[77] O. Hennigh. “Lat-Net: Compressing Lattice Boltzmann Flow Simulations using Deep Neural Networks”. In: *ArXiv e-prints* (May 2017). arXiv: 1705.09036 [stat.ML].

[78] Geoffrey Hinton et al. “Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups”. In: *Signal Processing Magazine, IEEE* 29.6 (2012), pp. 82–97.

[79] K. I. Hodges. “Feature Tracking on the Unit Sphere”. In: *Monthly Weather Review* 123.12 (1995), pp. 3458–3465. DOI: 10.1175/1520-0493(1995)123<3458:FTOTUS>2.0.CO;2. URL: https://doi.org/10.1175/1520-0493(1995)123%3C3458:FTOTUS%3E2.0.CO;2.

[80] Seungkyun Hong et al. “Globenet: Convolutional neural networks for typhoon eye tracking from remote sensing imagery”. In: *arXiv preprint arXiv:1708.03417* (2017).

[81] Ehsan Hosseini-Asl, Georgy Gimel’farb, and Ayman El-Baz. “Alzheimer’s Disease Diagnostics by a Deeply Supervised Adaptable 3D Convolutional Network”. In: (2016). eprint: arXiv:1607.00556.

[82] Itay Hubara et al. “Quantized Neural Networks: Training Neural Networks with Low Precision Weights and Activations”. In: *CoRR* abs/1609.07061 (2016).

[83] Forrest N. Iandola et al. “FireCaffe: near-linear acceleration of deep neural network training on compute clusters”. In: *CoRR* abs/1511.00175 (2015).

[84] Gilberto Iglesias, David C Kale, and Yan Liu. “An examination of deep learning for extreme climate pattern analysis”. In: *The 5th International Workshop on Climate Informatics*. 2015.

[85] *Insurance Journal*. https://www.insurancejournal.com/news/southcentral/2016/07/12/419831.htm.

[86] *Intel® distribution of Caffe**. https://github.com/intel/caffe. 2017.

[87] *Intel® Machine Learning Scaling Library for Linux* OS*. https://github.com/01org/MLSL. 2017.

[88] *Intel® Software Development Emulator*. https://software.intel.com/en-us/articles/intel-software-development-emulator. 2017.

[89] *Intergovernmental Panel on Climate Change, Fifth Assessment Report*. http://www.ipcc.ch/report/ar5.

[90] *Introducing DNN primitives in Intel® Math Kernel Library*. https://software.intel.com/en-us/articles/introducing-dnn-primitives-in-intelr-mkl. 2017.

[91] IPCC. *CMIP5 - Coupled Model Intercomparison Project Phase 5*. http://cmip-pcmdi.llnl.gov/cmip5/.

[92] *IPCC - Managing the Risks of Extreme Events and Disasters to Advance Climate Change Adaptation Report*. https://archive.ipcc.ch/report/srex/.

[93] Simon Jégou et al. "The one hundred layers tiramisu: Fully convolutional densenets for semantic segmentation". In: *Computer Vision and Pattern Recognition Workshops (CVPRW), 2017 IEEE Conference on*. IEEE. 2017, pp. 1175–1183.

[94] Soo Yeon Jeon et al. "Characterization of extreme precipitation within atmospheric river events over California". In: 2015.

[95] Shuiwang Ji et al. "3D convolutional neural networks for human action recognition". In: *IEEE transactions on pattern analysis and machine intelligence* 35.1 (2013), pp. 221–231.

[96] X. Jia et al. "Highly Scalable Deep Learning Training System with Mixed-Precision: Training ImageNet in Four Minutes". In: *ArXiv e-prints* (July 2018). arXiv: 1807.11205.

[97] Z. Kabluchko, M. Schlather, and L. de Haan. "Stationary max-stable fields associated to negative definite functions". In: *Annals of Probability* 37 (2009), pp. 2042–2065.

[98] Andrej Karpathy et al. "Large-scale video classification with convolutional neural networks". In: *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*. 2014, pp. 1725–1732.

[99] Anuj Karpatne and Vipin Kumar. "Big Data in Climate: Opportunities and Challenges for Machine Learning". In: *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. KDD '17. Halifax, NS, Canada: ACM, 2017, pp. 21–22. ISBN: 978-1-4503-4887-4. DOI: 10.1145/3097983.3105810. URL: http://doi.acm.org/10.1145/3097983.3105810.

[100] Will Kay et al. "The Kinetics Human Action Video Dataset". In: *arXiv preprint arXiv:1705.06950* (2017).

[101] Wesley Kendall et al. "Simplified parallel domain traversal". In: *Proceedings of 2011 International Conference for High Performance Computing, Networking, Storage and Analysis*. SC '11. Seattle, Washington: ACM, 2011, 10:1–10:11. ISBN: 978-1-4503-0771-0. DOI: 10.1145/2063384.2063397. URL: http://doi.acm.org/10.1145/2063384.2063397.

[102] Nitish Shirish Keskar et al. "On large-batch training for deep learning: Generalization gap and sharp minima". In: *arXiv:1609.04836* (2016).

[103] Viatcheslav V. Kharin et al. "Changes in temperature and precipitation extremes in the CMIP5 ensemble". In: *Climate Change* 119 (2013), pp. 345–357.

[104] Viatcheslav V. Kharin et al. "Changes in Temperature and Precipitation Extremes in the IPCC Ensemble of Global Coupled Model Simulations". In: *J. Climate* 20 (2007), pp. 1419–1444. DOI: 10.1175/JCLI4066.1.

[105] Diederik P. Kingma and Jimmy Ba. "Adam: A Method for Stochastic Optimization". In: *arXiv e-prints*, arXiv:1412.6980 (Dec. 2014), arXiv:1412.6980. arXiv: 1412.6980 [cs.LG].

[106] Diederik P Kingma and Jimmy Ba. "Adam: A method for stochastic optimization". In: *arXiv preprint arXiv:1412.6980* (2014).

[107] Diederik P Kingma and Max Welling. "Auto-encoding variational bayes". In: *arXiv preprint arXiv:1312.6114* (2013).

[108] Kenneth R. Knapp et al. "The International Best Track Archive for Climate Stewardship (IBTrACS)". In: *Bulletin of the American Meteorological Society* 91.3 (2010), pp. 363–376. DOI: 10.1175/2009BAMS2755.1. URL: http://dx.doi.org/10.1175/2009BAMS2755.1.

[109] T. R. Knutson et al. "Simulation of the Recent Multidecadal Increase of Atlantic Hurricane Activity Using an 18-km-Grid Regional Model". In: *Bulletin of the American Meteorological Society* 88.10 (2007), pp. 1549–1565.

[110] Thomas Knutson et al. "Tropical Cyclones and Climate Change Assessment: Part II. Projected Response to Anthropogenic Warming". In: *Bulletin of the American Meteorological Society* 0.0 (2019), null. DOI: 10.1175/BAMS-D-18-0194.1. eprint: https://doi.org/10.1175/BAMS-D-18-0194.1. URL: https://doi.org/10.1175/BAMS-D-18-0194.1.

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

[112] Kenneth E Kunkel et al. "Meteorological causes of the secular variations in observed extreme precipitation events for the conterminous United States". In: *Journal of Hydrometeorology* 13.3 (2012), pp. 1131–1141.

[113] Thorsten Kurth et al. "Exascale deep learning for climate analytics". In: *Proceedings of the International Conference for High Performance Computing, Networking, Storage, and Analysis*. IEEE Press. 2018, p. 51.

[114] Thorsten Kurth et al. "Deep Learning at 15PF: Supervised and Semi-supervised Classification for Scientific Data". In: *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*. SC '17. Denver, Colorado: ACM, 2017, 7:1–7:11. ISBN: 978-1-4503-5114-0. DOI: 10.1145/3126908.3126916. URL: http://doi.acm.org/10.1145/3126908.3126916.

[115] Thorsten Kurth et al. "Exascale Deep Learning for Climate Analytics". In: *Proceedings of the International Conference for High Performance Computing, Networking, Storage, and Analysis*. SC '18. Dallas, Texas: IEEE Press, 2018, 51:1–51:12. DOI: 10.1109/SC.2018.00054. URL: https://doi.org/10.1109/SC.2018.00054.

[116] Sally Langford, Samantha Stevenson, and David Noone. "Analysis of Low-Frequency Precipitation Variability in the CMIP5 Historical Simulations for Southwestern North America". In: *Journal of Climate* 27 (2014), pp. 2735–2756.

[117] Hugo Larochelle et al. "Exploring strategies for training deep neural networks". In: *The Journal of Machine Learning Research* 10 (2009), pp. 1–40.

[118] N.C Lau. "The stationary wave response to a tropical SST anomaly in an idealized GCM". In: *J. Armos. Sci.* 47 (1990), pp. 2546–2566.

[119] N.C Lau. "Variability of the observed midlatitude storm tracks in relation to low-frequency changes in the circulation pattern". In: *J. Armos. Sci.* 45 (1988), pp. 2718–2743.

[120] D. A. Lavers et al. "The detection of atmospheric rivers in atmospheric reanalyses and their links to British winter floods and the large-scale climatic circulation". In: *J. Geophys. Res.* 117 (2012), p. D20106. DOI: doi:10.1029/2012JD018027.

[121] David A. Lavers et al. "Future changes in atmospheric rivers and their implications for winter flooding in Britain". In: *Environmental Research Letters* 8.3 (2013), p. 034010. URL: http://stacks.iop.org/1748-9326/8/i=3/a=034010.

[122] David A Lavers et al. "The detection of atmospheric rivers in atmospheric reanalyses and their links to British winter floods and the large-scale climatic circulation". In: *Journal of Geophysical Research: Atmospheres* 117.D20 (2012).

[123] Andrew Lavin and Scott Gray. "Fast Algorithms for Convolutional Neural Networks". In: *CoRR* abs/1509.09308 (2015).

[124] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. "Deep learning". In: *nature* 521.7553 (2015), p. 436.

[125] L. R. Leung and Y. Qian. "Atmospheric rivers induced heavy precipitation and flooding in the western U.S. simulated by WRF regional climate model". In: *Geophys. Res. Lett.* 36 (2009). DOI: 10.1029/2008GL036445.

[126] Sergey Levine et al. “End-to-end training of deep visuomotor policies”. In: *The Journal of Machine Learning Research* 17.1 (2016), pp. 1334–1373.

[127] Chung-Yuan Lin, Sz-Yan Li, and Tsung-Han Tsai. “A scalable parallel hardware architecture for connected component labeling”. In: *ICIP*. IEEE, 2010, pp. 3753–3756. URL: http://dx.doi.org/10.1109/ICIP.2010.5653457.

[128] Wei Liu et al. “SSD: Single shot multibox detector”. In: *European Conference on Computer Vision*. Springer. 2016, pp. 21–37.

[129] Yunjie Liu et al. “Application of Deep Convolutional Neural Networks for Detecting Extreme Weather in Climate Datasets”. In: *CoRR* abs/1605.01156 (2016).

[130] Yunjie Liu et al. “Application of deep convolutional neural networks for detecting extreme weather in climate datasets”. In: *arXiv preprint arXiv:1605.01156* (2016).

[131] H. van Loon and J. C. Rogers. “The seesaw in winter temperatures between Greenland and northern Europe. Part I: General description”. In: *Mon. Wea. Rev.* 106 (1978), pp. 296–310.

[132] William Lotter, Greg Sorensen, and David Cox. “A multi-scale CNN and curriculum learning strategy for mammogram classification”. In: *Deep Learning in Medical Image Analysis and Multimodal Learning for Clinical Decision Support*. Springer, 2017, pp. 169–177.

[133] Jessica D. Lundquist et al. “Relationships between Barrier Jet Heights, Orographic Precipitation Gradients, and Streamflow in the Northern Sierra Nevada”. In: *Journal of Hydrometeorology* 11.5 (2010), pp. 1141–1156.

[134] L.J.P van der Maaten and G.E. Hinton. “Visualizing High-Dimensional Data Using t-SNE”. In: *Journal of Machine Learning Research* 9: 2579–2605 (Nov 2008).

[135] Ankur Mahesh et al. “Forecasting El Niño with Convolutional and Recurrent Neural Networks”. In: ().

[136] Ankur Mahesh et al. *Probabilistic Detection of Extreme Weather Using Deep Learning Methods*. Jan. 2019. URL: https://ams.confex.com/ams/2019Annual/webprogram/Paper354370.html.

[137] Alireza Makhzani et al. “Adversarial Autoencoders”. In: *CoRR* abs/1511.05644 (2015). URL: http://arxiv.org/abs/1511.05644.

[138] Martın Abadi et al. *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*. Software available from tensorflow.org. 2015. URL: https://www.tensorflow.org/.

[139] EC Massoud et al. “Global Climate Model Ensemble Approaches for Future Projections of Atmospheric Rivers”. In: *Earth's Future* 7.10 (2019), pp. 1136–1151.

[140] Amrita Mathuriya et al. “CosmoFlow: Using Deep Learning to Learn the Universe at Scale”. In: *Proceedings of SuperComputing*. 2018. arXiv: 1808.04728 [astro-ph.CO].

[141] Warren S McCulloch and Walter Pitts. "A logical calculus of the ideas immanent in nervous activity". In: *The bulletin of mathematical biophysics* 5.4 (1943), pp. 115–133.

[142] Amy McGovern et al. "Using artificial intelligence to improve real-time decision-making for high-impact weather". In: *Bulletin of the American Meteorological Society* 98.10 (2017), pp. 2073–2090.

[143] G. A. Meehl et al. "The WCRP CMIP3 multi-model dataset: A new era in climate change research". In: *Bulletin of the American Meteorological Society* 88 (2007), pp. 1383–1394.

[144] Ishan Misra, Abhinav Shrivastava, and Martial Hebert. "Watch and Learn: Semi-Supervised Learning of Object Detectors from Videos". In: *CoRR* abs/1505.05769 (2015). URL: http://arxiv.org/abs/1505.05769.

[145] Daniel Mitchell et al. "Half a degree additional warming, prognosis and projected impacts (HAPPI): background and experimental design". In: *Geoscientific Model Development* 10 (2017), pp. 571–583.

[146] Ioannis Mitliagkas et al. "Asynchrony begets momentum, with an application to deep learning". In: *arXiv:1605.09774* (2016).

[147] Adam H Monahan et al. "Empirical orthogonal functions: The medium is the message". In: *Journal of Climate* 22.24 (2009), pp. 6501–6514.

[148] G. Muszynski et al. "Topological data analysis and machine learning for recognizing atmospheric river patterns in large climate datasets". In: *Geoscientific Model Development* 12.2 (2019), pp. 613–628. DOI: 10.5194/gmd-12-613-2019. URL: https://www.geosci-model-dev.net/12/613/2019/.

[149] Vinod Nair and Geoffrey E Hinton. "Rectified linear units improve restricted boltzmann machines". In: *Proceedings of the 27th International Conference on Machine Learning (ICML)*. 2010, pp. 807–814.

[150] *National Weather Services*. https://www.weather.gov/hazstat/.

[151] P. Naveau et al. "Modeling pairwise dependence of maxima in space". In: *Biometrika* 96.1 (2009), pp. 1–17.

[152] *NCAR Command Language*. http://www.ncl.ucar.edu/.

[153] P. J. Neiman et al. "Diagnosis of an Intense Atmospheric River Impacting the Pacific Northwest: Storm Summary and Offshore Vertical Structure Observed with COSMIC Satellite Retrievals". In: *Monthly Weather Review* 136.11 (2008), pp. 4398–4420. DOI: 10.1175/2008MWR2550.1. eprint: http://journals.ametsoc.org/doi/pdf/10.1175/2008MWR2550.1. URL: http://journals.ametsoc.org/doi/abs/10.1175/2008MWR2550.1.

[154] P. J. Neiman et al. "Meteorological characteristics and overland precipitation impacts of atmospheric rivers affecting the West Coast of North America based on eight years of SSM/I satellite observations". In: *J. Hydrometeorol.* 9 (2008), pp. 22–47.

[155] *netCDF Operator*. http://nco.sourceforge.net/.

[156] Urs Neu and et al. “IMILAST: A Community Effort to Intercompare Extratropical Cyclone Detection and Tracking Algorithms”. In: *Bulletin of the American Meteorological Society* 94.4 (2013), pp. 529–547. DOI: 10.1175/BAMS-D-11-00154.1.

[157] Urs Neu et al. “IMILAST: A Community Effort to Intercompare Extratropical Cyclone Detection and Tracking Algorithms”. In: *Bulletin of the American Meteorological Society* 94.4 (2013), pp. 529–547. DOI: 10.1175/BAMS-D-11-00154.1. eprint: https://doi.org/10.1175/BAMS-D-11-00154.1. URL: https://doi.org/10.1175/BAMS-D-11-00154.1.

[158] Feng Niu et al. “Hogwild: A lock-free approach to parallelizing stochastic gradient descent”. In: *NIPS*. 2011, pp. 693–701.

[159] Mark Nixon and Alberto S. Aguado. *Feature Extraction in Computer Vision and Image Processing*. Oxford, UK: Newnes, 2002.

[160] NOAA. *ENSO Indices*. Oct. 2019. URL: https://www.weather.gov/fwd/indices.

[161] *NOAA National Centers for Environmental Information*. https://www.ncdc.noaa.gov/billions/.

[162] David S. Nolan and Michael G. McGauley. “Tropical cyclogenesis in wind shear: Climatological relationships and physical processes”. In: 2012.

[163] *NVIDIA Collective Communications Library (NCCL)*. URL: https://developer.nvidia.com/nccl.

[164] Paul A O’Gorman and John G Dwyer. “Using machine learning to parameterize moist convection: Potential for modeling of climate, climate change, and extreme events”. In: *Journal of Advances in Modeling Earth Systems* 10.10 (2018), pp. 2548–2563.

[165] Paul A O’Gorman and Tapio Schneider. “The physical basis for increases in precipitation extremes in simulations of 21st-century climate change”. In: *Proceedings of the National Academy of Sciences* 106.35 (2009), pp. 14773–14777.

[166] Travis A. O’Brien et al. “A fast and objective multidimensional kernel density estimation method: fastKDE”. In: *Computational Statistics  Data Analysis* 101 (2016), pp. 148–160. ISSN: 0167-9473. DOI: https://doi.org/10.1016/j.csda.2016.02.014.

[167] Travis A. O’Brien et al. “Reducing the computational cost of the ECF using a nuFFT: A fast and objective probability density estimation method”. In: *Computational Statistics  Data Analysis* 79 (2014), pp. 222–234. ISSN: 0167-9473. DOI: https://doi.org/10.1016/j.csda.2014.06.002.

[168] Travis A O’Brien et al. “A fast and objective multidimensional kernel density estimation method: fastKDE”. In: *Computational Statistics  Data Analysis* 101 (2016), pp. 148–160.

[169] Geert Jan van Oldenborgh et al. "Attribution of extreme rainfall from Hurricane Harvey, August 2017". In: *Environmental Research Letters* 12.12 (Dec. 2017), p. 124009. DOI: 10.1088/1748-9326/aa9ef2.

[170] R.K. Pachauri and A. Reisinger, eds. *Climate Change 2007: Synthesis Report – Contribution of Working Groups I, II and III to the Fourth Assessment Report of the Intergovernmental Panel on Climate Change*. Geneva, Switzerland: IPCC, 2007.

[171] *PAGODA: Parallel Analysis of Geoscience Data*. https://svn.pnl.gov/gcrm/wiki/Pagoda.

[172] Pardeep Pall, MR Allen, and Dáithı A Stone. "Testing the Clausius–Clapeyron constraint on changes in extreme precipitation under CO 2 warming". In: *Climate Dynamics* 28.4 (2007), pp. 351–363.

[173] *Parallel Climate Analysis Library*. http://trac.mcs.anl.gov/projects/parvis.

[174] *Paraview*. http://www.paraview.org/.

[175] Omkar M Parkhi, Andrea Vedaldi, and Andrew Zisserman. "Deep face recognition". In: *British Machine Vision Conference*. Vol. 1. 3. 2015, p. 6.

[176] Christina Patricola and Michael Wehner. "Anthropogenic influences on major tropical cyclone events". In: *Nature* 563 (Nov. 2018). DOI: 10.1038/s41586-018-0673-2.

[177] Ashley E. Payne and Gudrun Magnusdottir. "Dynamics of Landfalling Atmospheric Rivers over the North Pacific in 30 Years of MERRA Reanalysis". In: *Journal of Climate* 27 (2014), pp. 7133–7150.

[178] Ashley E Payne and Gudrun Magnusdottir. "An evaluation of atmospheric rivers over the North Pacific in CMIP5 and their response to warming under RCP 8.5". In: *Journal of Geophysical Research: Atmospheres* 120.21 (2015), pp. 11–173.

[179] *Piz Daint - Cray XC50, Xeon E5-2690v3 12C 2.6GHz, Aries interconnect, NVIDIA Tesla P100 | TOP500 Supercomputer Sites*.

[180] *Piz Daint | CSCS*. URL: https://www.cscs.ch/computers/piz-daint/.

[181] Prabhat. *Deep Learning for Science*. https://www.oreilly.com/ideas/a-look-at-deep-learning-for-science. 2017.

[182] Mr Prabhat et al. "TECA: Petascale Pattern Recognition for Climate Science". In: 9257 (Sept. 2015).

[183] Surendra Prabhat Byna et al. "TECA: Petascale pattern recognition for climate science". In: *International Conference on Computer Analysis of Images and Patterns*. Springer. 2015, pp. 426–436.

[184] Prabhat et al. "TECA: Petascale Pattern Recognition for Climate Science". In: *Computer Analysis of Images and Patterns*. Springer. 2015, pp. 426–436.

[185] Prabhat et al. "ClimateNet: an expert-labelled open dataset and Deep Learning architecture for enabling high-precision analyses of extreme weather". In: *Geophysical Model Development (In Review)* (2020).

[186] Prabhat et al. "TECA: A Parallel Toolkit for Extreme Climate Analysis". In: *ICCS* (2012).

[187] Prabhat et al. "TECA: A Parallel Toolkit for Extreme Climate Analysis". In: *Procedia Computer Science* 9.0 (2012). Proceedings of the International Conference on Computational Science, 2012, pp. 866–876. ISSN: 1877-0509. DOI: `http://dx.doi.org/10.1016/j.procs.2012.04.093`. URL: `http://www.sciencedirect.com/science/article/pii/S1877050912002141`.

[188] Prabhat et al. "TECA: Petascale Pattern Recognition for Climate Science". In: *CAIP* (2015).

[189] Evan Racah et al. "Semi-Supervised Detection of Extreme Weather Events in Large Climate Datasets". In: *arXiv:1612.02095* (2016).

[190] Evan Racah et al. "Extreme Weather: A large-scale climate dataset for semi-supervised detection, localization, and understanding of extreme weather events". In: *Advances in Neural Information Processing Systems 30*. 2017.

[191] Evan Racah et al. "ExtremeWeather: A large-scale climate dataset for semi-supervised detection, localization, and understanding of extreme weather events". In: Dec. 2017, pp. 3405–3416. URL: `https://papers.nips.cc/paper/6932-extremeweather-a-large-scale-climate-dataset-for-semi-supervised-detection-localization-and-understanding-of-extreme-weather-events`.

[192] Alexandar Radovic et al. "Machine Learning at the energy and intensity frontiers of particle physics". In: *Nature* (560 2018).

[193] F Martin Ralph et al. "Atmospheric rivers emerge as a global science and applications focus". In: *Bulletin of the American Meteorological Society* 98.9 (2017), pp. 1969–1973.

[194] F Martin Ralph, Paul J Neiman, and Richard Rotunno. "Dropsonde observations in low-level jets over the northeastern Pacific Ocean from CALJET-1998 and PACJET-2001: Mean vertical-profile and atmospheric-river characteristics". In: *Monthly weather review* 133.4 (2005), pp. 889–910.

[195] F. M. Ralph, P. J. Neiman, and R. Rotunno. "Dropsonde observations in low-level jets over the Northeastern Pacific Ocean from CALJET-1998 and CALJET-2001: Mean vertical-profile and atmospheric-river characteristics". In: *Mon. Weather Rev.* 133 (2005), pp. 889–910.

[196] F. M. Ralph, P. J. Neiman, and G. A. Wick. "Satellite and caljet aircraft observations of atmospheric rivers over the eastern north pacific ocean during the winter of 1997/98". In: *Mon. Weather Rev.* 132 (2004), pp. 1721–1745.

[197] F. M. Ralph et al. “Flooding on California’s Russian River: Role of atmospheric rivers”. In: *Geophys. Res. Lett.* 33 (2006). DOI: 10.1029/2006GL026689.

[198] F. Martin Ralph, Paul J. Neiman, and Gary A. Wick. “Satellite and CALJET Aircraft Observations of Atmospheric Rivers over the Eastern North Pacific Ocean during the Winter of 1997/98”. In: *Monthly Weather Review* 132.7 (2004), pp. 1721–1745.

[199] F. Martin Ralph et al. “A Multiscale Observational Case Study of a Pacific Atmospheric River Exhibiting Tropical-Extratropical Connections and a Mesoscale Frontal Wave”. In: *Monthly Weather Review* 139.4 (2011), pp. 1169–1189. DOI: 10.1175/2010MWR3596.1. URL: http://journals.ametsoc.org/doi/abs/10.1175/2010MWR3596.1.

[200] Antti Rasmus et al. “Semi-supervised learning with ladder networks”. In: *Advances in Neural Information Processing Systems*. 2015, pp. 3546–3554.

[201] Joseph Redmon et al. “You Only Look Once: Unified, Real-Time Object Detection”. In: *CoRR* abs/1506.02640 (2015). URL: http://arxiv.org/abs/1506.02640.

[202] Joseph Redmon et al. “You only look once: Unified, real-time object detection”. In: *CVPR*. 2016, pp. 779–788.

[203] Jeffrey Regier, Prabhat, and Jon McAuliffe. “A deep generative model for astronomical images of galaxies”. In: *NIPS Workshop: Advances in Approximate Bayesian Inference*. 2015.

[204] *Remote Sensing Systems Special Sensor Microwave Imager instrument*. http://www.remss.com/missions/ssmi.

[205] Shaoqing Ren et al. “Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks”. In: (2015). eprint: arXiv:1506.01497.

[206] Shaoqing Ren et al. “Faster r-cnn: Towards real-time object detection with region proposal networks”. In: *NIPS*. 2015, pp. 91–99.

[207] Mark D. Risser and Michael F. Wehner. “Attributable Human-Induced Changes in the Likelihood and Magnitude of the Observed Extreme Precipitation during Hurricane Harvey”. In: *Geophysical Research Letters* 44.24 (2017), pp. 12, 457–12, 464. DOI: 10.1002/2017GL075888. URL: https://agupubs.onlinelibrary.wiley.com/doi/abs/10.1002/2017GL075888.

[208] Frank Rosenblatt. “The perceptron: a probabilistic model for information storage and organization in the brain.” In: *Psychological review* 65.6 (1958), p. 386.

[209] DE Ruhmelhart, GE Hinton, and RJ Wiliams. “Learning representations by back-propagation errors”. In: *Nature* 323 (1986), pp. 533–536.

[210] Olga Russakovsky et al. “Imagenet large scale visual recognition challenge”. In: *International Journal of Computer Vision* 115.3 (2015), pp. 211–252.

[211] Olga Russakovsky et al. *Beyond ImageNet Large Scale Visual Recognition Challenge*. 2017. URL: http://image-net.org/challenges/beyond_ilsvrc.php (visited on 05/12/2017).

[212] Bryan C Russell et al. "LabelMe: a database and web-based tool for image annotation". In: *International journal of computer vision* 77.1-3 (2008), pp. 157–173.

[213] Jonathan J. Rutz, W. James Steenburgh, and F. Martin Ralph. "Climatological Characteristics of Atmospheric Rivers and Their Inland Penetration over the Western United States". In: *Monthly Weather Review* 142 (2014), pp. 905–921. URL: http://dx.doi.org/10.1175/MWR-D-13-00168.1.

[214] Tim Salimans et al. "Improved Techniques for Training GANs". In: (2016). eprint: arXiv:1606.03498.

[215] *Scaling Deep Learning on 18,000 GPUs*. https://www.nextplatform.com/2017/03/28/scaling-deep-learning-beyond-18000-gpus/. 2017.

[216] M. Schlather. "Models for stationary max-stable random fields". In: *Extremes* 5 (2002), pp. 33–44.

[217] M. Schlather and J. A. Tawn. "A dependence measure for multivariate and spatial extreme values: Properties and Inference". In: *Biometrika* 90.1 (2003), pp. 139–156.

[218] Alexander Sergeev and Mike Del Balso. "Horovod: fast and easy distributed deep learning in TensorFlow". In: *arXiv preprint arXiv:1802.05799* (2018).

[219] Pierre Sermanet et al. "OverFeat: Integrated Recognition, Localization and Detection using Convolutional Networks". In: *International Conference on Learning Representations (ICLR)*. 2014.

[220] Xingjian Shi et al. "Convolutional LSTM network: A machine learning approach for precipitation nowcasting". In: *Advances in Neural Information Processing Systems: Twenty-Ninth Annual Conference on Neural Information Processing Systems (NIPS)*. 2015.

[221] C. A. Shields et al. "Atmospheric River Tracking Method Intercomparison Project (ARTMIP): Project Goals and Experimental Design". In: *Geoscientific Model Development Discussions* 2018 (2018), pp. 1–55. DOI: 10.5194/gmd-2017-295. URL: https://www.geosci-model-dev-discuss.net/gmd-2017-295/.

[222] C. A. Shields et al. "Atmospheric River Tracking Method Intercomparison Project (ARTMIP): project goals and experimental design". In: *Geoscientific Model Development* 11.6 (2018), pp. 2455–2474. DOI: 10.5194/gmd-11-2455-2018. URL: https://www.geosci-model-dev.net/11/2455/2018/.

[223] Christine A Shields et al. "Atmospheric River Tracking Method Intercomparison Project (ARTMIP): project goals and experimental design". In: *Geoscientific Model Development* 11.6 (2018), pp. 2455–2474.

[224] J. Sillmann et al. “Climate extremes indices in the CMIP5 multimodel ensemble: Part 1. Model evaluation in the present climate”. In: *J. Geophys. Res. Atmos.* 118 (2013), pp. 1716–1733. DOI: 10.1002/jgrd.50203.

[225] J. Sillmann et al. “Climate extremes indices in the CMIP5 multimodel ensemble: Part 2. Future climate projections”. In: *J. Geophys. Res. Atmos.* 118 (2013), pp. 2473–2493. DOI: 10.1002/jgrd.50188.

[226] K. Simonyan and A. Zisserman. “Very Deep Convolutional Networks for Large-Scale Image Recognition”. In: *Internaltional Conference on Learning Representation (ICLR)*. 2015.

[227] R. L. Smith. “Max-stable processes and spatial extremes”. Unpublished manuscript. 1990.

[228] Jasper Snoek, Hugo Larochelle, and Ryan P Adams. “Practical bayesian optimization of machine learning algorithms”. In: *Advances in neural information processing systems*. 2012, pp. 2951–2959.

[229] J. T. Springenberg et al. “Striving for Simplicity: The All Convolutional Net”. In: *International Conference on Learning Representation (ICLR)*. 2015.

[230] Jost Tobias Springenberg. “Unsupervised and semi-supervised learning with categorical generative adversarial networks”. In: *arXiv preprint arXiv:1511.06390* (2015).

[231] Nitish Srivastava, Elman Mansimov, and Ruslan Salakhutdinov. “Unsupervised learning of video representations using lstms”. In: *CoRR, abs/1502.04681* 2 (2015).

[232] Luigi di Stefano and Andrea Bulgarelli. “A Simple and Efficient Connected Components Labeling Algorithm”. In: *ICIAP ’99: Proceedings of the 10th International Conference on Image Analysis and Processing*. Washington, DC, USA: IEEE Computer Society, 1999, p. 322.

[233] Karsten Steinhaeuser, Nitesh Chawla, and Auroop Ganguly. “Comparing predictive power in climate data: Clustering matters”. In: *Advances in Spatial and Temporal Databases* (2011), pp. 39–55.

[234] *Summit - IBM Power System AC922, IBM POWER9 22C 3.07GHz, NVIDIA Volta GV100, Dual-rail Mellanox EDR Infiniband | TOP500 Supercomputer Sites*. URL: https://www.top500.org/system/179397.

[235] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. “Sequence to sequence learning with neural networks”. In: *Advances in neural information processing systems*. 2014, pp. 3104–3112.

[236] Kenji Suzuki, Isao Horiba, and Noboru Sugie. “Linear-time connected-component labeling based on sequential local operations”. In: *Comput. Vis. Image Underst.* 89.1 (2003), pp. 1–23.

[237] Daniel L Swain et al. “Increasing precipitation volatility in twenty-first-century California”. In: *Nature Climate Change* 8.5 (2018), pp. 427–433.

[238] Christian Szegedy et al. "Going deeper with convolutions". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2015, pp. 1–9.

[239] *TensorFlow website*. URL: https://tensorflow.org.

[240] Benjamin A Toms, Karthik Kashinath, Da Yang, et al. "Deep learning for scientific inference from geophysical data: The Madden-Julian oscillation as a test case". In: *arXiv preprint arXiv:1902.04621* (2019).

[241] Du Tran et al. "Learning Spatiotemporal Features with 3D Convolutional Networks". In: (2014). eprint: arXiv:1412.0767.

[242] John Tsitsiklis, Dimitri Bertsekas, and Michael Athans. "Distributed asynchronous deterministic and stochastic gradient optimization algorithms". In: *IEEE transactions on automatic control* 31.9 (1986), pp. 803–812.

[243] Jasper RR Uijlings et al. "Selective search for object recognition". In: *International Journal of Computer Vision* 104.2 (2013), pp. 154–171.

[244] Paul A Ullrich and Colin M Zarzycki. "TempestExtremes: a framework for scale-insensitive pointwise feature tracking on unstructured grids". In: *Geoscientific Model Development* 10.3 (2017), p. 1069.

[245] *Ultrascale Visualization - Climate Data Analysis Tools*. http://uv-cdat.llnl.gov/.

[246] Gabriel Vecchi et al. "Tropical cyclone sensitivities to CO2 doubling: Roles of atmospheric resolution, synoptic variability and background climate changes". In: *Climate Dynamics* 53 (Nov. 2019), pp. 5999–6033. DOI: 10.1007/s00382-019-04913-y.

[247] *VisIt visualization tool*. https://wci.llnl.gov/codes/visit/.

[248] F Vitart, JL Anderson, and WF Stern. "Impact of large-scale circulation on tropical storm frequency, intensity, and location, simulated by an ensemble of GCM integrations". In: *Journal of Climate* 12.11 (1999), pp. 3237–3254.

[249] F Vitart, JL Anderson, and WF Stern. "Simulation of interannual variability of tropical storm frequency in an ensemble of GCM integrations". In: *Journal of Climate* 10.4 (1997), pp. 745–760.

[250] Kevin Walsh and Ian G Watterson. "Tropical cyclone-like vortices in a limited area model: comparison with observed climatology". In: *Journal of Climate* 10.9 (1997), pp. 2240–2259.

[251] Kevin Walsh et al. "The Tropical Cyclone Climate Model Intercomparison Project". In: *Hurricanes and Climate Change: Volume 2*. Ed. by James B. Elsner et al. Dordrecht: Springer Netherlands, 2010, pp. 1–24. ISBN: 978-90-481-9510-7. DOI: 10.1007/978-90-481-9510-7_1. URL: https://doi.org/10.1007/978-90-481-9510-7_1.

[252] KJE Walsh et al. "Objectively determined resolution-dependent threshold criteria for the detection of tropical cyclones in climate models and reanalyses". In: *Journal of Climate* 20.10 (2007), pp. 2307–2314.

[253] J.-X. Wang et al. “A Comprehensive Physics-Informed Machine Learning Framework for Predictive Turbulence Modeling”. In: *ArXiv e-prints* (Jan. 2017). arXiv: 1701.07102 [physics.flu-dyn].

[254] Kuang-Bor Wang et al. “Parallel Execution of a Connected Component Labeling Operation on a Linear Array Architecture”. In: *Journal of Information Science And Engineering* 19 (2003), pp. 353–370.

[255] S-Y Simon Wang et al. “Quantitative attribution of climate effects on Hurricane Harvey’s extreme rainfall in Texas”. In: *Environmental Research Letters* 13.5 (Apr. 2018), p. 054014. DOI: 10.1088/1748-9326/aabb85. URL: https://doi.org/10.1088%2F1748-9326%2Faabb85.

[256] Xiaolan L. Wang and Yang Feng. “Inter-comparison of extra-tropical cyclone activity in eight reanalysis datasets”. In: *EGU General Assembly Research Abstract* (2014).

[257] Michael D. Warner, Clifford F. Mass, and Eric P. Salathé Jr. “Wintertime Extreme Precipitation Events along the Pacific Northwest Coast: Climatology and Synoptic Evolution”. In: *Monthly Weather Review* 140 (2012), pp. 2021–2043.

[258] Michael D Warner, Clifford F Mass, and Eric P Salathé Jr. “Changes in winter atmospheric rivers along the North American west coast in CMIP5 climate models”. In: *Journal of Hydrometeorology* 16.1 (2015), pp. 118–128.

[259] *WCRP Coupled Model Intercomparison Project Phase 6*. http://www.wcrp-climate.org/wgcm-cmip/wgcm-cmip6.

[260] P. J. Webster et al. “Changes in Tropical Cyclone Number, Duration, and Intensity in a Warming Environment”. In: *Science* 309.5742 (2005), pp. 1844–1846.

[261] M. F. Wehner et al. “The effect of horizontal resolution on simulation quality in the Community Atmospheric Model, CAM5.1”. In: *Journal of Advances in Modeling Earth Systems* 6 (2014), pp. 980–997.

[262] M. F. Wehner et al. “Towards Direct Simulation of Future Tropical Cyclone Statistics in a High-Resolution Global Atmospheric Model”. In: *Advances in Meteorology*. Vol. 2010. 2010, p. 915303. DOI: 10.1155/2010/915303. URL: http://www.hindawi.com/journals/amet/2010/915303/.

[263] Michael F. Wehner. “Very extreme seasonal precipitation in the NARCCAP ensemble: Model performance and projections”. In: *Climate Dynamics* 40 (2013), pp. 59–80.

[264] Michael F Wehner et al. “The effect of horizontal resolution on simulation quality in the Community Atmospheric Model, CAM5. 1”. In: *Journal of Advances in Modeling Earth Systems* 6.4 (2014), pp. 980–997.

[265] Michael F Wehner et al. “Changes in tropical cyclones under stabilized 1.5 and 2.0° C global warming scenarios as simulated by the Community Atmospheric Model under the HAPPI protocols”. In: *Earth System Dynamics* 9.1 (2018).

[266] Michael Wehner et al. "Resolution Dependence of Future Tropical Cyclone Projections of CAM5.1 in the U.S. CLIVAR Hurricane Working Group Idealized Configurations". In: *Journal of Climate* 28.10 (2015), pp. 3905–3925. DOI: 10.1175/JCLI-D-14-00311.1.

[267] Daphna Weinshall, Gad Cohen, and Dan Amir. "Curriculum learning by transfer learning: Theory and experiments with deep networks". In: *arXiv:1802.03796* (2018).

[268] G. B. Weller, D. S. Cooley, and S. R. Sain. "An investigation of the pineapple express phenomenon via bivariate extreme value theory". In: *Environmetrics* 23 (2012), pp. 420–439.

[269] William F. Whitney et al. "Understanding Visual Concepts with Continuation Learning". In: (2016). eprint: arXiv:1602.06822.

[270] Gary A. Wick, Paul J. Neiman, and F. Martin Ralph. "Description and Validation of an Automated Objective Technique for Identification and Characterization of the Integrated Water Vapor Signature of Atmospheric Rivers". In: *IEEE Transactions on Geoscience and Remote Sensing* 51.4 (2013), pp. 2166–2176.

[271] Kesheng Wu, Ekow Otoo, and Kenji Suzuki. "Optimizing Two-pass Connected component Labeling Algorithms". In: *Pattern Anal. Appl.* 12.2 (Feb. 2009), pp. 117–135. ISSN: 1433-7541. DOI: 10.1007/s10044-008-0109-y.

[272] Kesheng Wu et al. "Finding regions of interest on toroidal meshes". In: *Computational Science & Discovery* 4.1 (2011), p. 015003. URL: http://stacks.iop.org/1749-4699/4/i=1/a=015003.

[273] Jianwen Xie, Song-Chun Zhu, and Ying Nian Wu. "Synthesizing Dynamic Textures and Sounds by Spatial-Temporal Generative ConvNet". In: (2016). eprint: arXiv:1606.00972.

[274] Shi Xingjian et al. "Convolutional LSTM network: A machine learning approach for precipitation nowcasting". In: *Advances in Neural Information Processing Systems*. 2015, pp. 802–810.

[275] Li Yao et al. "Describing videos by exploiting temporal structure". In: *Proceedings of the IEEE International Conference on Computer Vision*. 2015, pp. 4507–4515.

[276] Y. You, I. Gitman, and B. Ginsburg. "Large Batch Training of Convolutional Networks". In: *ArXiv e-prints* (Aug. 2017). arXiv: 1708.03888 [cs.CV].

[277] Amir R Zamir et al. "Taskonomy: Disentangling task transfer learning". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2018, pp. 3712–3722.

[278] Ce Zhang and Christopher Re. "DimmWitted: A Study of Main-Memory Statistical Analytics". In: *PVLDB* 7.12 (2014), pp. 1283–1294.

[279] Sixin Zhang, Anna Choromanska, and Yann LeCun. “Deep learning with Elastic Averaging SGD”. In: *CoRR* abs/1412.6651 (2014). arXiv: 1412.6651. URL: http://arxiv.org/abs/1412.6651.

[280] Yuting Zhang, Kibok Lee, and Honglak Lee. “Augmenting Supervised Neural Networks with Unsupervised Objectives for Large-scale Image Classification”. In: *arXiv preprint arXiv:1606.06582v1* (2016).

[281] Junbo Zhao et al. “Stacked what-where auto-encoders”. In: *arXiv arXiv:1506.02351* (2015).

[282] Y. Zhu and R. E. Newell. “A Proposed Algorithm for Moisture Fluxes from Atmospheric Rivers”. In: *Monthly Weather Review - USA* 126.3 (1998), pp. 725–735.

[283] Yong Zhu and Reginald E. Newell. “Atmospheric Rivers and Bombs”. In: *Geophysical Research Letters* 21.18 (1994), pp. 1999–2002.