UNIVERSITY OF CALIFORNIA,
IRVINE

Artificial Neural Network Impact on Cloud Parameterization and Land-Atmosphere Interactions

THESIS

submitted in partial satisfaction of the requirements
for the degree of

MASTER OF SCIENCE

in Mathematical, Computational, and Systems Biology

by

Galen Yacalis

Thesis Committee:
Assistant Professor Mike Pritchard, Chair
Chancellor's Professor James Randerson
Chancellor's Professor John Lowengrub

2018

DEDICATION

To

my fiancée, family, and friends

in recognition of their unending love and support

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGMENTS

I would like to express the deepest appreciation to my committee chair, Assistant Professor Mike Pritchard. His enthusiasm is infectious, and discussing research with him is energizing. His guidance, encouragement, and faith in me has been a tremendously positive influence, and I am profoundly grateful.

I would like to thank my committee member Chancellor’s Professor John Lowengrub, who has always been my greatest advocate in pursuing my interests.

I would like to thank my committee member Chancellor’s Professor James Randerson, whose both direct and indirect impact on me have been a great source of inspiration and admiration.

In addition, a huge thank you to Pierre Gentine for his tireless efforts and support. Another thank you to Stephan Rasp both for being a friend and possessing an intellect and mastery to chase after. And lastly, a big thank you to all of my colleagues in the Earth System Science department and Mathematical, Computational, and Systems Biology program at the University of California Irvine; none of my efforts would have been possible absent their great assistance, advice, and encouragement.

The author acknowledges the Texas Advanced Computing Center (TACC) at The University of Texas at Austin for providing HPC resources that have contributed to the research results reported within this paper. URL: http://www.tacc.utexas.edu

ABSTRACT OF THE THESIS

Artificial Neural Network Impact on Cloud Parameterization and Land-Atmosphere Interactions

By

Galen Yacalis

Master of Science in Mathematical, Computational, and Systems Biology

University of California, Irvine, 2018

Assistant Professor Mike Pritchard, Chair

Ecosystem dynamics are heavily dependent on atmospheric inputs such as rainfall, and are in turn an integral part of land-atmosphere coupling and the global carbon cycle. These global interactions and cycles are commonly modeled by Earth System Models (ESMs), and one of the largest sources of uncertainty in current models is cloud simulation techniques.

Superparameterization (SP) is a proven method to better resolve cloud and rainfall processes in ESMs, but it is prohibitively computationally expensive for long-term climate simulations. The first part of this thesis suggests that – although the latest advances in manycore supercomputing do not provide a promising solution to this problem – a neural network (NN) can successfully be trained on an SP version of the Community Atmosphere Model (CAM) to emulate SP behavior at a fraction of the cost. Incorporating the NN into CAM under idealized aquaplanet conditions results in the Neural Network Community Atmosphere Model (NNCAM).

For NNCAM to work in fully comprehensive models that include interactive vegetation, deeper tests are needed. The second and most major contribution of this thesis is thus to analyze the one-way coupling of NNCAM onto the Community Land Model (CLM). Gross primary productivity (GPP) and net ecosystem exchange (NEE) from ensembles of one-way atmospheric forcing onto a CLM grid cell in the Amazon Basin for NNCAM and SP-CAM are shown to be statistically different from CAM but not each other. Additionally, results suggest that mean precipitation is the largest contributing factor to GPP and NEE in the Amazon Basin.

INTRODUCTION

Background

Ecosystem dynamics are a critical part of land-atmosphere interactions and climate dynamics. Land cover and vegetation type have a huge impact on global energy fluxes, water balance, and the carbon cycle, with complex interactions and feedback effects (Stephens et al., 2012; Miralles et al., 2018). Forests in particular have disproportionally large nonlinear effects on the hydrologic and carbon cycles through biological and physical atmospheric interactions, especially wet tropical forests (Bonan, 2008). Beyond presence and absence, biodiversity, genetic diversity, and functional species grouping have consequential climate impacts (Thompson et al., 2009; McMahon et al., 2011).

Energy Fluxes

As energy from the Sun in the form of radiation warms the Earth, the flow of energy among the land, ocean, and atmosphere determines the planet's climate behavior. This energy flow includes heat, moisture, chemical concentration, and radiation exchange (Jung et al., 2011).

The energy flux between the land and atmosphere is determined by patterns of absorbed and reflected solar and longwave radiation, conduction, convection, and latent heat transference. All matter emits radiation, and the wavelength of that radiation is determined by the object's temperature. Solar radiation is also known as shortwave radiation because hotter objects emit radiation at shorter, more energetic wavelengths. Radiation emitted by the soil, atmosphere, vegetation, or any other mass on Earth is known as longwave radiation because these lower temperatures objects emit longer, less energetic wavelengths.

Briefly, all heat on Earth is originally derived from solar radiation(see Figure 0.1). At each interaction with the atmosphere, land, ocean, and ice, solar radiation is either reflected or absorbed. The surface property measuring how much solar radiation is reflected by a surface is albedo. Dark objects absorb more radiation and have low albedo, whereas bright objects such as ice have high albedo. As radiation from the sun enters the Earth's atmosphere, it either is absorbed, is reflected, or passes through the air and clouds. When the solar radiation reaches a solid object such as ice, land, vegetation, concrete, and water, a portion is reflected back to the atmosphere and another portion is absorbed by the object's surface. The reflected solar radiation again is absorbed by, reflected back by, or passes through the air and clouds. Longwave radiation emitted by the Earth's surface and atmosphere follows a similar process; however, rate of absorption and reflection of radiation is determined in part by wavelength and the extent of other absorbing and emitting bodies along the wavelength's path, and thus the rates of absorption and reflection for short and longwave radiation are not the same.

Conduction and convection transfer energy from the Earth's surface to the atmosphere in the form of heat. Hotter air near the surface is less dense that the air above it and rises, carrying the heat upwards. When water vapor is transported to the atmosphere from plant transpiration or evaporation, it also carries latent heat. Latent heat is the energy required to cause water to change phases, from a liquid to a gas. When water vapor is carried upwards into the atmosphere, it takes that energy with it, and when it condenses again into liquid form the energy is released in the form of heat. Because of this, plant transpiration, determined by stomatal resistance, water and nutrient availability, soil properties, and other factors, plays an important role in heat transference from the land to the atmosphere through water vapor fluxes.

The energy budget of the climate refers to the net absorption or reflection of heat by radiation. In a closed system, net heat is neither gained nor lost. For the Earth's climate, this is generally true, but if the Earth absorbs more radiation from the sun than it reflects back to space, the overall climate gets warmer. As may be apparent, the growth and decay of different varieties of vegetation affect land surface albedo, which influences the energy budget of the climate; and the climate in turn influences the growth and decay of vegetation.

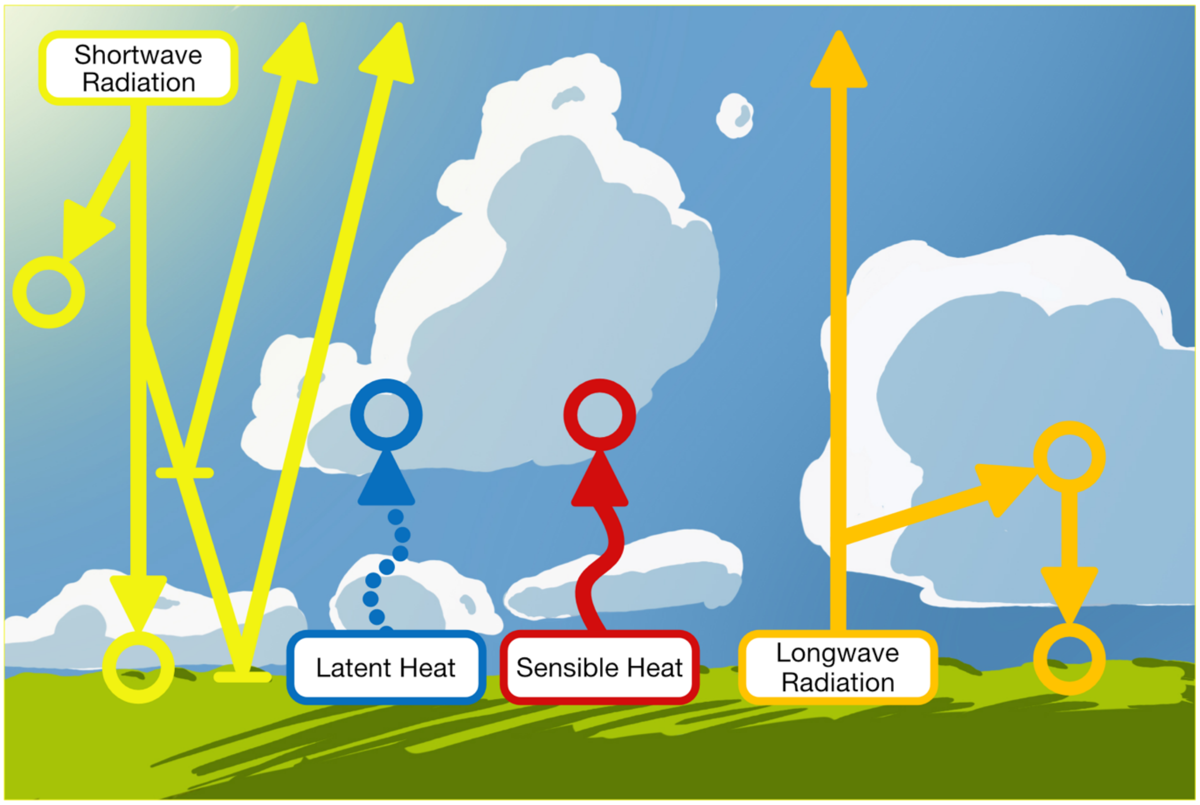


Figure I1: Overview of the energy flux of the Earth. Circles indicate instances of heat or radiation absorption. Solar radiation is absorbed or reflected at each interaction with matter, from the atmosphere to the surface and back. Longwave radiation from the Earth either escapes to beyond the atmosphere or is trapped by clouds and atmospheric gases to be distributed back to the surface. Sensible heat diffuses upwards from the surface. Latent heat is transported through water vapor and released during cloud condensation. Description of energy fluxes and figure content adapted from Kiehl et al., 1997; Trenberth et al., 2009; Stephens et al., 2012; and Wofsy et al., 2007.

Hydrologic Cycle

The hydrologic cycle is the cycling of water among the land, ocean, ice, and atmosphere. Liquid water from the soil or ocean is warmed enough to change phases and become water vapor. When enough of the water vapor accumulates around aerosols in clouds, precipitation results. Over vegetative land surfaces, some precipitation is intercepted by leaves and branches and quickly evaporates. Throughfall is the amount of precipitation that passes through plant canopies and reaches the surface. Throughfall is absorbed into the soil until soil saturation is reached, and the remaining water becomes runoff. Soil moisture near the surface evaporates, while deeper soil moisture is held up to a maximum concentration that is determined by the soil type.

Plant effects on soil moisture are numerous. Some plant roots have been shown to redistribute soil water from lower, more moist regions to regions near the surface (Bonan, 2015). However, a far greater impact by plants is the evaporation of soil moisture through stomata while they are open during photosynthesis, a process known as transpiration.

Typically the processes of transpiration and of evaporation from the soil and canopy are combined into a single parameter, evapotranspiration. Overall, evapotranspiration heavily affects local climate, as the land cycles about 60% of precipitation it receives back to the atmosphere (Chen et al., 2013; Jung et al., 2010; Oki & Kanae, 2006).

Water is required for photosynthesis, which, using trees as an example, occurs in the leaves. Water that lands on leaves is not available for use by the plant and can quickly evaporates. The water that trees use for both photosynthesis and other cellular processes is drawn from the soil. Photosynthesis transforms water from the soil and carbon dioxide from the atmosphere into sugar and oxygen. For vascular plants, the process of transpiration transports water from roots through xylem to the leaves, and makes it available for use during photosynthesis. As the leaf stomata open to absorb carbon from the atmosphere for use in photosynthetic chemical reactions, some of the water in the leaf becomes exposed and transpires to the atmosphere, completing one possible path of the hydrologic cycle. Globally, land evapotranspiration is dominated by transpiration (Lawrence et al., 2007). Plants affect the evaporation of water and availability of precipitation via transpiration, and precipitation impacts soil moisture availability for plants to use for photosynthesis.

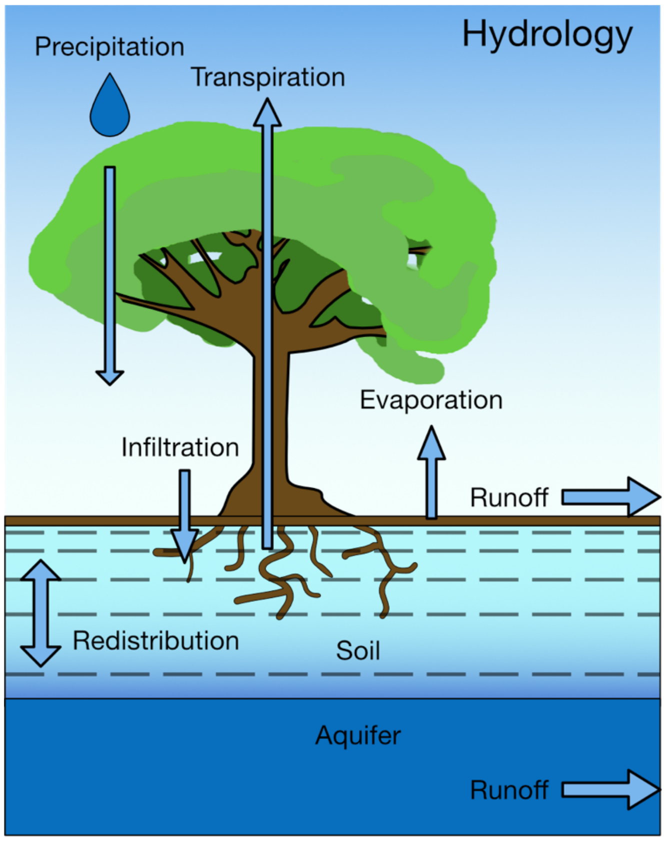


Figure I2: The active hydrology of Community Land Model. Precipitation from the atmosphere evaporates or infiltrates into the soil. Soil water diffuses until the soil is saturated, and further precipitation results in runoff. Heavy episodes of precipitation can cause upper soil layers to saturate before moisture can diffuse into unsaturated lower soil layers, increasing runoff compared to lighter episodes of the same quantity of precipitation (Bonan, 2015). Water uptake from vegetation moves from the roots to the leaves and escapes into the atmosphere as water vapor through the process of transpiration. Description of hydrologic cycle and figure adapted from Lawrence et al., 2011, and Bonan, 2015.

Carbon Cycle

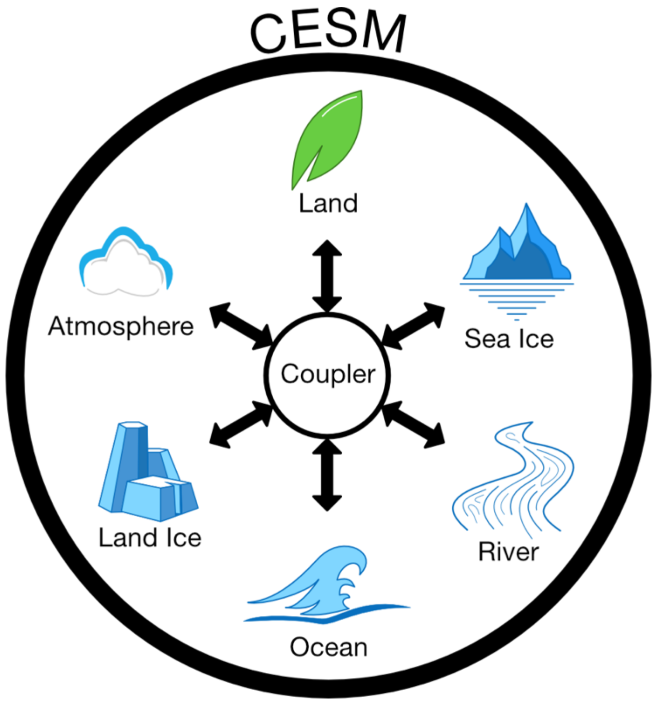
One of the largest factors on the energy budget is the composition of the atmosphere. For example, most harmful shortwave radiation is absorbed or reflected by the ozone layer, without which the surface would be bombarded and life could not exist as it currently does (Solomon, 1999). Re-radiation back to Earth of longwave radiation is influenced by the presence of greenhouse gasses, one of the most important of which is carbon dioxide. The specific concentration of carbon dioxide in the atmosphere is determined by geological processes, anthropogenic activity, and biological processes. The carbon fluxes of geological processes and anthropogenic activity (combined approximately $10PgCyr^{-1}$) are small compared to biological processes. In particular, terrestrial plant carbon flux via photosynthesis, respiration, growth, decay, and destruction by natural events is estimated to be 142.5 $PgCyr^{-1}$ (Bonan, 2015). Changes in the terrestrial plant contribution to the carbon cycle could have large impacts on the acceleration of climate change and species distribution (Cox et al., 2000; Sellers et al., 1990).

The measure of net flux of carbon between the atmosphere and biosphere is net ecosystem exchange (NEE). It includes fire, land use, harvest, maintenance respiration deficit, and gross primary production (GPP), which is calculated as photosynthesis plus autotrophic respiration. If NEE is positive then the land is considered a carbon source, as the amount of carbon leaving the land exceeds the amount absorbed or fixed; if NEE is negative, the land is a carbon sink. Forests hold a large majority of above-ground biomass, and the Amazon rainforest in particular contributes to a large portion of both forest biomass and GPP (Chambers et al., 2000; Thompson et al., 2009).

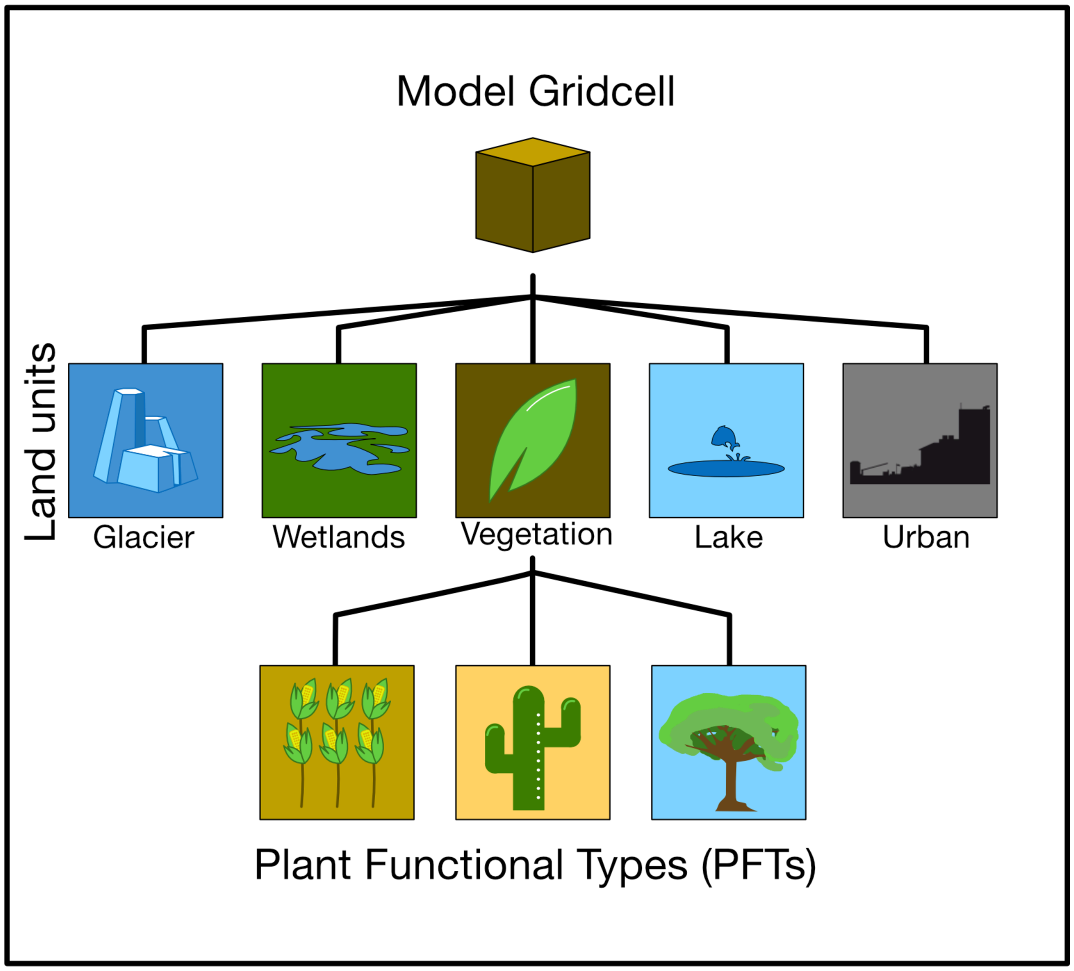
CHAPTER 1

Earth System Models and Superparameterization

Climate simulations are performed by numerical models with multiple millions of lines of code, written by hundreds of scientists, and are referred to by many names, but the term preferred here will be Earth System Model (ESM). Worldwide there are dozens of independent ESMs, such as those used for international IPCC assessment, each with their own configurable set of equations, parameters, and modeled processes. The most widely used ESM worldwide is the Community Earth System Model (CESM). CESM is modular and able to use a combination of smaller models that are specific to aspects of the Earth system: atmosphere, land, sea ice, ocean, land ice, and rivers. A coupler acts to allow the models to interact among themselves for complex climate simulations.



(a)



(b)

Figure 1.1: (a) Overview of Community Earth System Model (CESM) components and (b) components of an Earth System Model (ESM) grid cell. CESM has a coupler acting between 6 models, simulating the atmosphere, land, ocean, rivers, land ice, and sea ice. An ESM grid cell has a vertical column extending from the Earth's surface to the stratosphere; on land, it is fractionally a distinct set of land types, not all of which are included in the figure. Vegetation is divided into multiple plant functional types (PFTs), each with different properties.

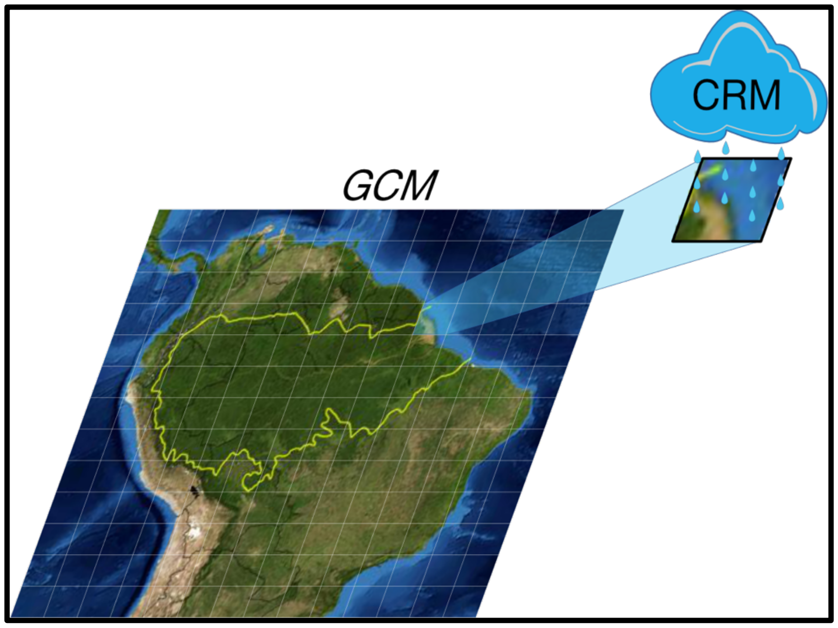
The atmospheric model used in the experiments of this paper is a legacy version of the Community Atmosphere Model (CAM). Broadly speaking, the behavior of the atmosphere in CAM is governed by the Navier-Stokes equations, which describe the motion of a fluid; in the case of the Earth's atmosphere, the fluid is a mixture of gases located in spherical coordinates and is moving across a rotating sphere. The equations govern the conservation of mass, momentum in three dimensions, and energy. The full set of equations are not solvable and are numerically approximated using physical scaling laws relevant to the affordable grid resolution of the numerical calculation (50-100 km horizontal resolution is typical today).

The largest cause of simulation spread among atmospheric models, including CAM, is the parameterization of cloud feedbacks (Flato et al., 2013). Clouds are parameterized because the physics involved in cloud formation occur at spatial scales that are much smaller than the minimum grid scale that a global atmospheric model can afford to resolve on current supercomputers. For global simulations, CAM and other models within CESM operate on grid cells that are typically 50-200 kilometers in width; convective processes, boundary layer turbulence, microphysics, entrainment, and other significant atmospheric processes would require grid cells of only a few hundreds to thousands of meters across, adding great computational complexity (Parishani et al., 2017).

Cloud resolving models (CRMs) have attempted to address this issue. ESMs typically contain atmospheric grid columns at low resolution which parameterize cloud processes. The atmosphere is simulated as if clouds were not present, and after each timestep, atmospheric variables are adjusted based on cloud process parameterizations that rely on limiting assumptions about the missing physics. In contrast, CRMs replace the parameterizations in the low resolution grid column with a finer, convective-permitting resolution subgrid that more explicitly simulates cloud processes at a shorter timescale, such as deep convection, boundary layer turbulence, and condensation, although some parameterizations still exist (Randall et al., 2003). Superparameterization (SP) is the nomenclature for the integration of a CRM into an ESM to create such a “multi-scale modeling framework”.

One of the hallmarks of SP is that it typically simulates precipitation extremes better than conventionally parameterized atmospheric global models like CAM (Koopermanet al., 2016); this allows it to better simulate soil moisture variability and the hydrologic cycle, as the atmospheric component is the most uncertain piece of land-atmosphere interactions that impact precipitation (Sun & Pritchard, 2016). Upper precipitation extremes can increase the level of runoff during rainstorms by supplying water faster that the infiltration rate of the soil (infiltration-excess) (Bonan, 2015), thereby decreasing the overall deep soil moisture in those areas compared to the same precipitation quantity during a rainstorm in which infiltration-excess runoff is absent. In ecosystems such as the Amazon, deep soil moisture is required by certain plants to maintain photosynthesis during the dry season (Allen et al., 2010), so the impact of rainfall events by SP on deep soil moisture can potentially increase or decrease plant survivability and greatly impact rainforest population dynamics based around precipitation throughfall and infiltration-excess. The consequence of SP’s impact on soil moisture and soil temperature can also influence the occurrence of fires, which greatly affects the vegetative composition and biodiversity of ecosystems (Thonicke et al., 2001). Another hallmark is that its ability to better resolve GCM subgrid-scale convective heating and moistening produces more realistic simulations of mesoscale processes such as the Madden-Juilan oscillation, an important intraseasonal climate pattern that dominates tropical variability and impacts global climate (Benedict & Randall, 2009; Thayer-Calder & Randall, 2009).

(a)



(b)

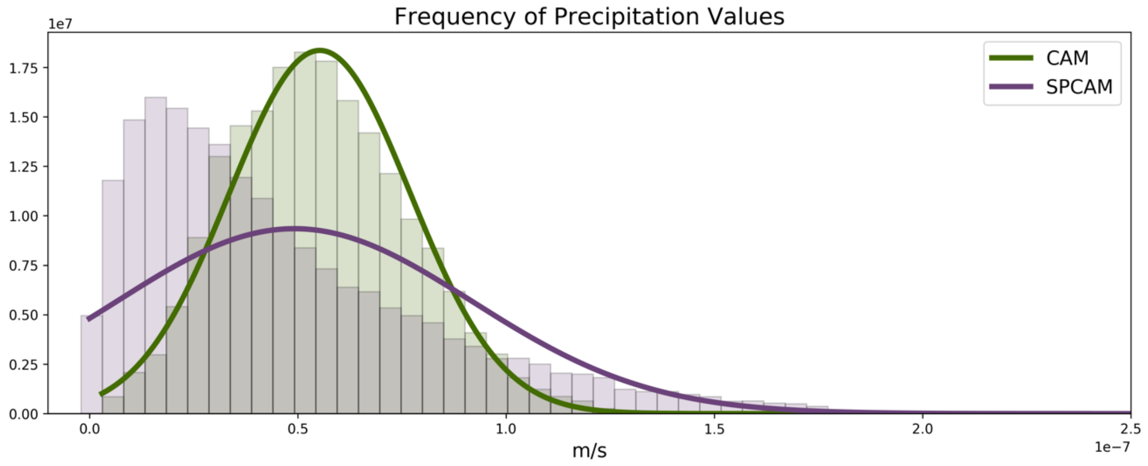


Figure 1.2: (a) depicts an embedded CRM within an ESM grid cell. The CRM has a higher resolution subgrid that calculates atmospheric processes separate from ESM and at shorter time intervals. At each ESM timestep, the CRM modifies the ESM's atmospheric state based on calculations occurring since the previous ESM timestep. (b) displays a histogram of the

frequency of precipitation values averaged over 176 grid cells for 1 year simulations from a parameterized ESM (CAM) and an ESM with embedded CRMs (superparameterized CAM, SPCAM). SPCAM's finer resolution allows it to capture more precipitation extremes than CAM, particularly upper precipitation extremes.

The integration of SP into regular CAM involves replacing the conventional cloud parameterization with superparameterization, and is called SPCAM. As previously mentioned, the thousands of CRMs embedded under the SP approach do have limitations in complexity; the CRM arrays are only 2D and not of 3D, and still parameterize microphysics, radiation, and small scale turbulence. Accepting this, currently the main disadvantage of SPCAM over CAM is that it is 1-2 orders of magnitude more computationally expensive than conventional parameterization (Randall et al., 2003). A symptom of this problem is that no major national climate prediction center, such as those contributing to the IPCC, has yet to adopt the SP method for mainstream climate science. Advances in computing technology have made longer SPCAM climate simulations more feasible, but the refactoring of the underlying CRM code to fully utilize new hardware technology is not automatic. The next section presents an example of efforts to utilize more recent hardware to reduce the computational limits of using SP.

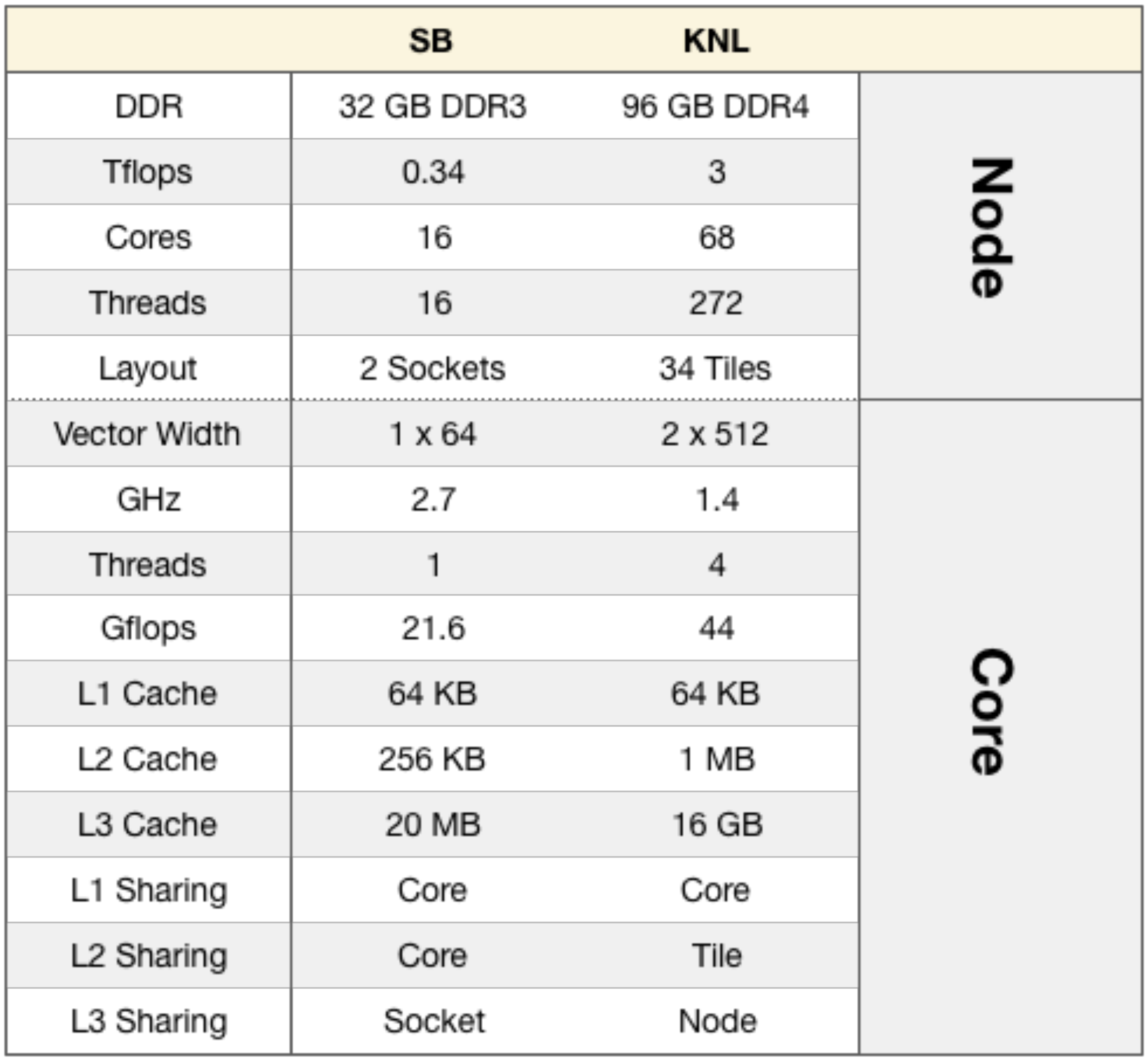
CHAPTER 2

Exploring the Potential of Intel's Many-Core Technology to Accelerate SP Simulations

To alleviate some of the limitations of computational complexity inherent to SP, leveraging recent advances in massively parallel high performance computing (HPC) technology via model optimization is a clear strategy worth pursuing. Currently SP simulations are limited to a maximum core count of $\sim$ 10k physical processors (a limit set by the number of embedded CRMs, or horizontal grid columns) of the host planetary scale model. As supercomputers change from fast-chip / low core count nodes to slow-chip / manycore nodes, it becomes important to explore the potential for a further decomposition of the CRMs towards making use of petascale HPC resources. This is most readily achieved through experimentation with offline experiments using the same CRM that is usually embedded within SP; this CRM is called The System for Atmospheric Modeling (SAM) (Khairoutdinov & Randall, 2003).

A project was thus initiated to assess the parallel scaling properties of SAM version 6.10.6 on the next-generation Intel architecture Knight's Landing (KNL). The HPC cluster used for the tests was Stampede2 at The Texas Advanced Computing Center (TACC), which designs and operates a number of different supercomputing clusters. Stampede2 entered full production in 2017 and is comprised exclusively of KNL nodes (Texas Advanced Computing Center, 2018). In addition to measuring the parallel scaling efficiency of SAM on KNL compared to its predecessor system at TACC, the Sandy Bridge (SB) cluster, the project entailed an assessment of the model's performance bottlenecks and performed a suite of sensitivity tests that attempted to improve the model's performance by better exploiting the hardware capabilities of KNL.

KNL breaks tradition from previous generation multicore nodes by introducing many-core nodes (Jeffers et al., 2016a). The distinction between SB and KNL is that instead of the 16 high performance cores per SB node, KNL implements nodes comprised of 68 cores with approximately half the processing power, arranged into tiles designed for faster memory sharing than previous generation hardware in order to take advantage of modern coding practices of vectorization and parallelization. The multicore architecture of SB splits cores into 2 sockets, with half of the cores being placed in one socket and the second half in the other socket; this has different memory consequences than the tile arrangement of KNL for vectorization and parallelization. A few of the most basic differences between the architectures are listed in Table 2.1.



	SB	KNL	
DDR	32 GB DDR3	96 GB DDR4	Node
Tflops	0.34	3	
Cores	16	68	
Threads	16	272	
Layout	2 Sockets	34 Tiles	
Vector Width	1 x 64	2 x 512	Core
GHz	2.7	1.4	
Threads	1	4	
Gflops	21.6	44	
L1 Cache	64 KB	64 KB	
L2 Cache	256 KB	1 MB	
L3 Cache	20 MB	16 GB	
L1 Sharing	Core	Core	
L2 Sharing	Core	Tile	
L3 Sharing	Socket	Node	

Table 2.1: A sample of differences between Sandy Bridge and Knight's Landing. KNL has more memory and is capable of more threads, larger vectors, and better flops per watt. SB has a higher processing power, 2.7 GHz per core versus KNL's 1.4 GHz per core. Unless software leverages the advantages of KNL, SB's better processing power will overtake KNL during performance tests.

Differences in memory and thread number account for some of the largest and most important changes from a multicore to many-core architecture. Caches are small chunks of memory space designed for fast reading and writing of data; the order of caches from fastest to slowest in SB and KNL is L1, L2, and L3. Beyond L3 cache, memory management takes place on slower, but higher memory capacity disk space, DDR. Compared to SB's multicore architecture, which has small L1-L3 caches and almost always must interface with DDR for memory storage, KNL has a large L3 cache (16GB) that is shared within a node that, when properly leveraged, can boost performance drastically for processes that need less than 16GB of memory (Jeffers et al., 2016b).

Regarding L1 and L2 cache-sharing, the basic difference between architectures is shown in Figure 2.1. In KNL, two cores (a single tile) share L2 cache. This can greatly boost performance of tasks if the memory for a process can be shared among 8 or less threads.

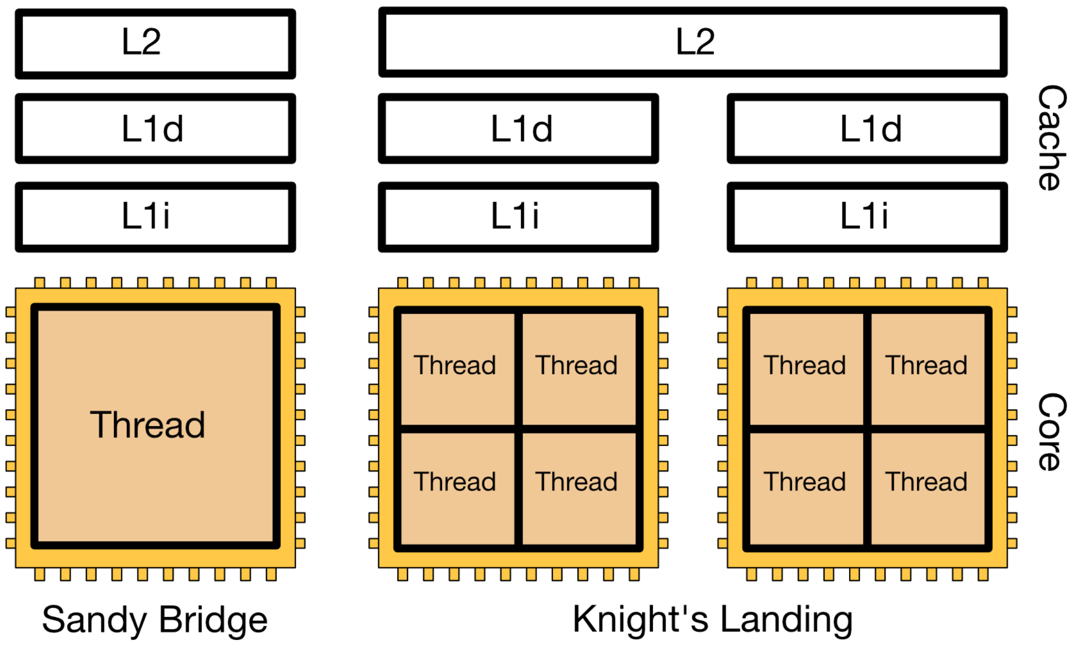


Figure 2.1: A display of the fundamental memory difference between a single Sandy Bridge core and a Knight's Landing tile with two cores. SB's core is only capable of a single thread, and has small local L1 and L2 cache. By contrast, each of KNL's cores on a tile is capable of 4

threads which all share L1 cache, and the 8 threads of the tile share L2 cache. If memory and threading is managed properly, huge performance gains can be made on KNL hardware.

There was reason to believe that KNL could make major improvements to SAM performance over SB. Tests conducted by Jeffers et al. 2016 on KNL effects on another cloud-resolving atmospheric model code, the Weather Research and Forecasting Model (WRF), a multipurpose numerical weather prediction system, showed that KNL performed better than previous generation processors because of its higher memory bandwidth, larger thread counts, and better vector capabilities (Jeffers et al., 2016c). The system of parallelization in WRF is similar to SAM in that it has the capacity for decomposition through 2 different parallelization regimes, one for thread parallelism and one through task parallelism. Given these similarities between WRF and SAM, it is logical that similar performance gains might be accessible for augmented SP on modern hardware. In practice, however, since all codes suffer unique bottlenecks, this is important to test.

I tested two SAM versions, one only using a 1-level task parallelization regime, and the other using a 2-level task and thread parallelization regime. Basic wallclock tests were performed. Initial conditions for all tests were identical, and the single-moment microphysics configuration of SAM was used. SB can incorporate a coprocessor for its processes, but it was not utilized in the tests performed. SAM documentation suggests that different domain sizes may have a large impact on performance (Khairoutdinov, 2014), so the tests were performed for a variety of domain sizes; however, the overall behavior across domain sizes was the same in all cases, and only one for the single-level decomposition case is shown in the Figure 2.2.

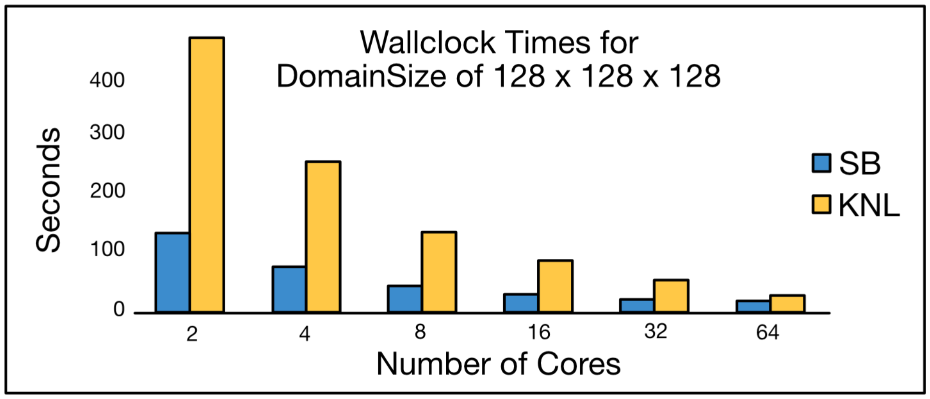
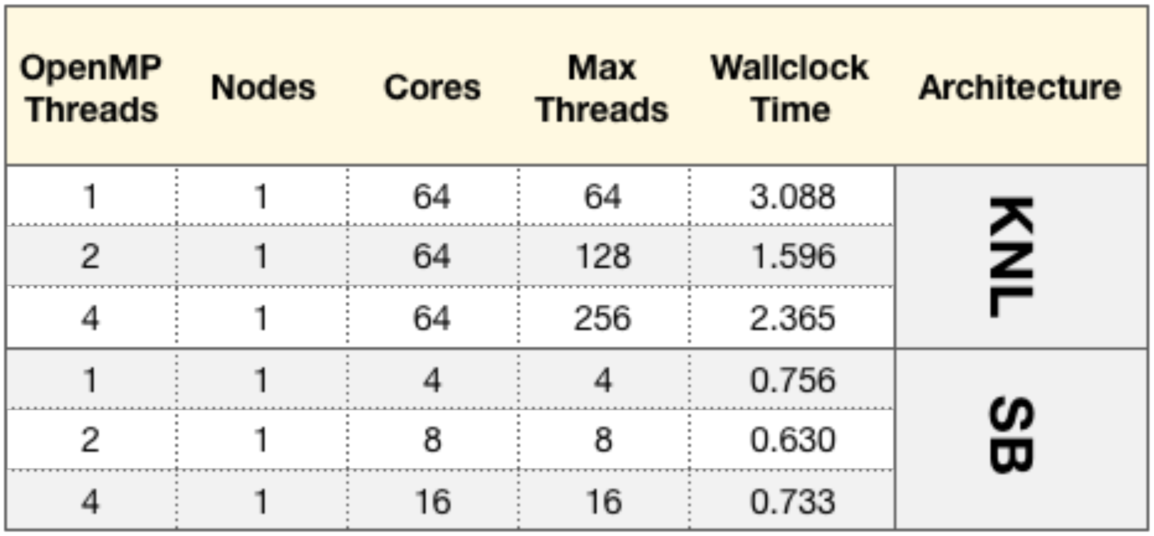


Figure 2.2: Wallclock times for a SB and KNL SAM simulation lasting 30 timesteps under task-level parallelism. The better processing power of SB cores outperforms KNL even when using more cores. This is due to KNL's architecture not being fully utilized with regard to parallelization, vectorization, and memory sharing.

As could be expected, when using only single level decomposition KNL performed worse on a core-to-core basis than SB, which has more powerful individual cores: two SB cores performed better than two KNL cores, and 64 SB cores on four nodes performed better than 64 cores on one KNL node. However, results also showed that KNL performed worse even on a node-to-node basis—64 KNL cores on one KNL node performed worse than 16 SB cores on one SB node.

The obvious conclusion to be made is that the version of SAM used, with task-level parallelism but not thread-level, did not fully take advantage of KNL's main strength, which is in parallelizable code. Therefore the expectation could be made that performance of SAM on a KNL node would be improved once 2-level parallelization was more fully incorporated.

Looking at the most time-intensive processes of the code, which were recorded by SAM timing reports, the most wallclock time was spent on pressure and radiation processes. Parallelizing loops in the underlying Fortran code of those processes was expected to be the most effective method to test potential improvements, while making as little code changes as possible. This would result in the 2-level parallelization scheme similar to that of WRF. The results can be seen in Table 2.2.



OpenMP Threads	Nodes	Cores	Max Threads	Wallclock Time	Architecture
1	1	64	64	3.088	KNL
2	1	64	128	1.596	
4	1	64	256	2.365	
1	1	4	4	0.756	SB
2	1	8	8	0.630	
4	1	16	16	0.733	

Table 2.2: Wallclock comparison of the computation time span per timestep between Sandy Bridge and Knight's Landing for a 128x1x128 domain. OpenMP is a programming paradigm used for threading. Each test utilized only one node. Because the test was designed to compare SB and KNL on a thread-to-thread basis, and SB has one thread per core, the number of SB cores used in each test was required to be equal to the number of threads used. KNL benefits more from utilizing multiple threads than SB, but its overall

time is still slower—the fastest threading regime on KNL is slower than the slowest threading regime on SB.

From Table 2.2 it can be seen that SAM was still slower on KNL than SB even after an additional parallelization technique was used. The table shows results for only a 128x1x128 domain, but every domain size tested had the same relative performance in that SB was always faster than KNL. Code changes were made in for-loops of pressure and radiation processes to utilize threading; utilizing multiple threads on a single instance of a for-loop meant that multiple iterations could be executed simultaneously. However, not enough memory was shared among threads for the greater memory sharing capabilities of KNL to have a large advantage over SB. The conclusion was that KNL could not be properly leveraged without significant refactoring of the underlying SAM code. Further investigation revealed that the changes would have to be of a non-trivial, fundamental nature to see great improvement.

The largest issue for performance on KNL is that SAM subdomains are explicitly core-mapped (Khairoutdinov, 2014). The number of cores to be used for a simulation are required to be specified before code compilation, and is intended for core-level parallelism, not thread-level parallelism. This makes sense given that SAM was originally programmed to be parallelizable on hardware systems that were widespread when the code was originally written in the early 2000's. On SB, single cores are very powerful and threading is not as beneficial as it is on KNL. Even implementing a far more extensive thread-level parallelization regime than was tested would be limited in its effectiveness on KNL: mapping each subdomain to a single core precludes the main advantage of KNL, which is allowing multithreading among cores for memory-sharing intensive tasks, and especially on tiles. Properly leveraging KNL would require SAM code to be refactored to allow for subdomains to be thread-mapped across multiple KNL tiles, which would fundamentally change the coding paradigm.

Other issues for implementing SAM on KNL exist as well. One method for discovering the best domain sizes on KNL is to fix the cores used and vary the domain size (Jeffers et al., 2016).; the obvious problem is that often in an experiment the domain size is very intentional. Additionally, new hardware architectures are being developed at a constant rate, and refactoring SAM for KNL use might quickly become outdated as new technology is introduced, such as TACC's new Skylake cluster (Texas Advanced Computing Center, 2018), or more advanced GPUs. These issues may be the reason Intel has decided to discontinue the hardware KNL is based on. Other groups of scientists in the HPC community have had similar problems of requiring large amount of refactoring for performance gains on KNL (Trader, 2018).

CHAPTER 3

Artificial Neural Networks

In Chapter 1 we have reviewed the conceptual and philosophical advantages of explicit cloud-resolving atmospheric simulation for improving the accuracy of biosphere simulations and their climate feedbacks. Leveraging modern hardware and software improvements to make SP less computationally cumbersome is thus essential. While Chapter 2 has shown that hardware advances do not provide a practical solution without major code refactoring, a more relatively heretofore unexplored methodology is the integration of machine learning techniques into climate simulations. This chapter will review proof-of-concept work in which software engineering work done by me as part of this thesis led to new machine learning tools resulting in a publication on which I am a co-author (Gentine et al., 2018) in which an artificial neural network (abbreviated here as simply neural network, NN) is successfully trained on SPCAM data to emulate the basic physics of SP within a CAM simulation at a fraction of the computational cost.

What is a neural network? Broadly speaking, a simple NN is an ordered series of sets of two matrices; each set consists of a "weight" parameters matrix, a "bias" parameters matrix, and one or more functions serving to introduce an element of non-linearity into the series ("activation functions"). For basic NNs, the weights and biases of the series are initialized at specified values and then systematically updated with the goal of arriving at values that, through multiplication, addition, and mutual linkage through assumed nonlinear activation functions, can discretely approximate even very high dimensional nonlinear functions (Fausett, L. V., 1994).

Updating the parameters of the NN (which is a model) is referred to as training. As the model is trained, it gradually "learns" the values that the parameters need to take in order to best approximate the desired nonlinear function.

Supervised learning is one form of training a NN can undertake (Chapelle, 2006). The model receives a series of inputs one by one with the goal of reproducing a matching set of target outputs. As it is trained, the model systematically and gradually alters its parameters to most closely match the target output when a given a particular input. How well the model performs is sometimes called its "fit". The method by which the fit is evaluated is through a function known as the cost function, a hyperparameter of the model; for each input the cost function measures the error between the model's output and the target output. The parameters are then systematically updated through a process known as back-propagation. This network update is intended for the next training instance's model output to better match the next target output and the result of the cost function to be reduced. In effect, supervised learning is the process by which the value of the cost function of a NN is minimized.

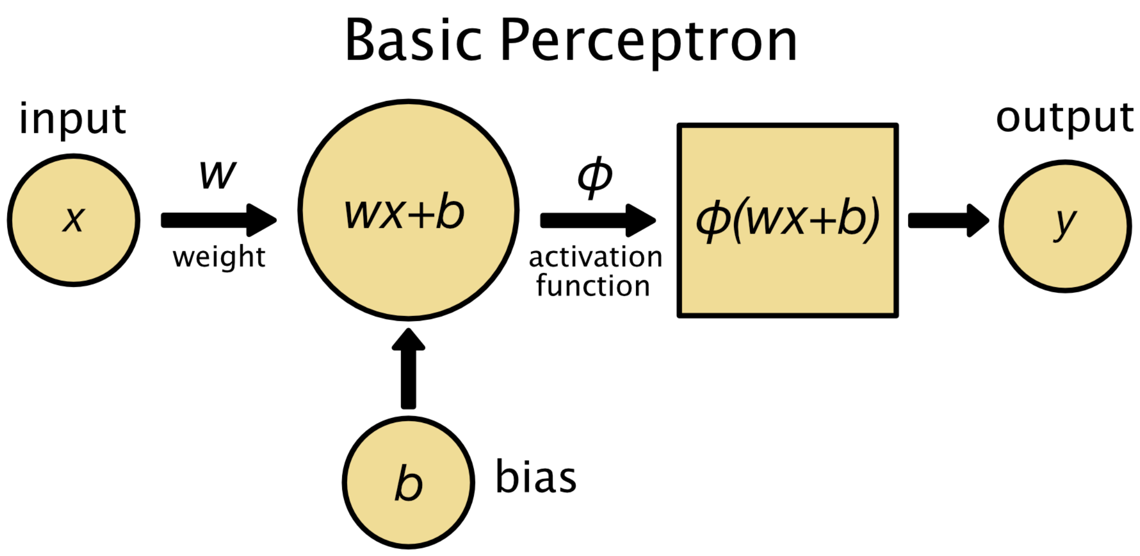


Figure 3.1: A basic perceptron, also called a neuron or node, one of the building blocks of modern neural networks (Kawaguchi, 2000). The perceptron receives input *x*, which is multiplied by a weight *w*. A bias *b* is added to the value to produce an intermediary variable; the intermediary variable becomes the input to the activation function ϕ, which produces $\hat{y}$ as the final output of the entire perceptron. Deep neural networks are formed from multiple interconnected neurons (Schalkoff, 1997). The equation for the output of a single perceptron with a single input is: $\hat{y} = \phi(wx + b)$.

The simplest individual component of a NN is the perceptron, also called a neuron. In a NN that consists of only a single neuron, such as in Figure 3.1, the process follows this procedure:

(1) The perceptron receives an input value.

(2) That value is multiplied by a parameter called a weight.

(3) Then a separate parameter value is added, called a bias.

(4) The resulting value is modified by a function, called the activation function.

(5) The output value of the activation function becomes the output of the neuron.

A stack of neurons is called a layer, and a layer can have any arbitrary number of neurons. If multiple layers are stacked, then a multi-layer NN is created, originally called multilayer perceptrons (Kawaguchi, 2000), seen in Figure 3.2. Theoretically, any nonlinear function can be approximated by just a single-layer NN; the theory is known as the Universal Approximation Theorem (Schmidhuber, 2015). More specifically, the theory states that a compact subset of any continuous function can be approximated below a specified error bound by a finite number of neurons in a single-layer neural network if the activation function of the neurons is continuous, monotonically increasing, non-constant, and bounded (such as a sigmoid function). While single-layer networks have infinite modeling power, they may require a near-infinite number of neurons per layer. In practice, deeper layered networks are used with sigmoid or non-sigmoid activation functions, as they have been empirically shown in many circumstances to produce better results than shallower networks (Wang & Raj, 2017).

The parameters of a neural network that get trained are its weights and biases. The final values of the weights and biases, along with network architecture and hyperparameters, are what determines the skill of the NN in approximating a function. When a training sample is given to a NN, it takes the input and produces a prediction output, $\hat{y}$. The prediction $\hat{y}$ is compared to the target output y through the cost function J and a loss value is produced. Then each weight and bias gets updated through a process known as backpropagation, which implements a form of stochastic gradient descent (SGD) (Nielsen, 2015). In SGD, the gradient of the loss function with respect to each parameter is calculated, and each parameter is then individually modified by its calculated gradient.

Once all training samples have been used in this manner, a single “epoch” has been completed, and the process repeats. This process repeats until either the loss reaches a specified minimum value or a maximum number of epochs or time is reached.

The previous explanation of SGD is also known as online SGD, because the weights and biases are updated after every training example. In minibatch SGD, the gradient of the loss function with respect to all the weights and biases is calculated for each training example in the minibatch; then, after all training samples in minibatch have been used, the weights and biases are updated with the average of the gradients, multiplied by the learning rate, η.

As previously mentioned, for many applications deeper networks tend to be more effective than shallow networks, and more neurons will improve the NN’s accuracy (Glorot & Bengio, 2010). However, there is always a point at which any model can become too complex—in these cases, the model “overfits” the training data and does not generalize well when given new information. Since the goal of training is to perform well on future, unknown data (i.e. potentially noise-corrupted or out-of-sample), achieving the correct overall level of complexity and number of total parameters is a crucial task. If the network is too simple, it will perform poorly on both training and test data (underfitting); if it is too complex, it may perform well on training data but generalize poorly when given test data (overfitting).

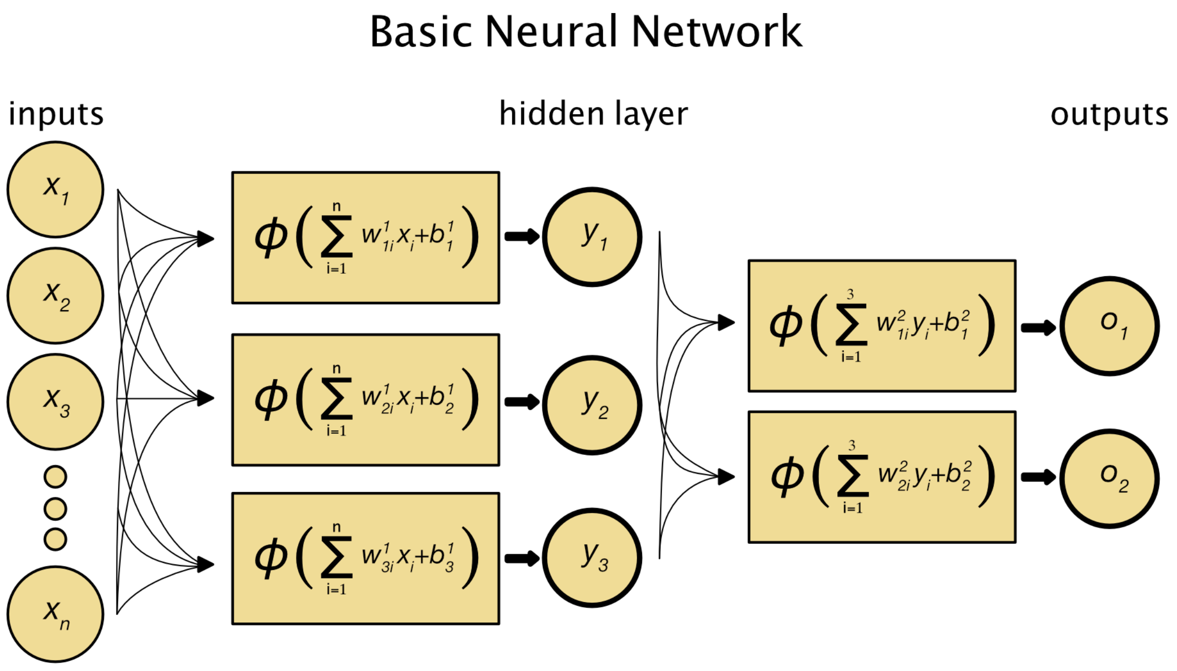

Basic Neural Network
inputs
hidden layer
outputs
x_1
x_2
x_3
x_n
$\phi\left(\sum_{i=1}^{n} w^1_{1i}x_i+b^1_1\right)$
$\phi\left(\sum_{i=1}^{n} w^1_{2i}x_i+b^1_2\right)$
$\phi\left(\sum_{i=1}^{n} w^1_{3i}x_i+b^1_3\right)$
y_1
y_2
y_3
$\phi\left(\sum_{i=1}^{3} w^2_{1i}y_i+b^2_1\right)$
$\phi\left(\sum_{i=1}^{3} w^2_{2i}y_i+b^2_2\right)$
o_1
o_2

Figure 3.2: The basic outline of a neural network. This NN is comprised of a single hidden layer with 3 neurons. A distinct set of weights and biases exist between the input layer and hidden layer, and between the hidden layer and the output layer. The value of neuron *j* in the hidden layer is: $y_j = \phi\left(\sum_{i=1}^{n} w_{ji}x_i + b_j\right)$, where n is the number of inputs, ϕ is an activation function, w_{ji} is the value of the weight between the ith input and the jth neuron in the hidden layer, x_i is the value of the input at the ith index, and b_j is the bias value for the jth neuron in the hidden layer. During training, weights and biases are updated. The powerful capabilities of NNs have had many different applications in a variety of disciplines (Karpathy et al., 2014; Maddison et al., 2014; Zeiler et al., 2013).

Hypothesis: Our hypothesis was that it might be possible to train an NN on SPCAM data via supervised learning and then use the NN within a CAM simulation as a cloud parameterization step instead of using SP or traditional CAM parameterization; ideally, the NN would perform so well that its effects would be similar to SP, but be much faster. The training phase of the NN would require a computational burden, but after training, the computational cost of using the NN would be the same during the simulation no matter how complex a NN was used, and potentially even slightly less costly than CAM's original, non-SP parameterization step. This idea that the physical coupled partial differential equations of a cloud resolving model simulation, representing conservation of momentum, energy and turbulent chaos (i.e. Navier-Stokes, scaled for atmospheric moist convection), as well as approximated turbulent diffusion and cloud microphysics, might be replaceable with a NN is controversial but its promise will be demonstrated in this chapter.

The difference between using the NN as a cloud parameterization step within an atmospheric simulation and using CAM's original cloud parameterization is that the latter represents incomplete attempts made by scientists through theory and empirical cloud process effects observed in the atmosphere through a limited deterministic model that makes considerable assumptions (about unresolved cloud geometry, assumed equilibrium behaviors, etc); by contrast, the NN parameter values are systematically trained through supervised learning to best approximate SP cloud processes, which through their explicit character are capable of a much more diverse set of responses than conventional parameterizations owing to their realistic complexity and lack of such assumptions. An important distinction to make is that strictly speaking, the intention during the training of the NN is not to realistically parameterize cloud processes; it is instead to best approximate the effects of SP, which we already recognize to realistically represent them. Functionally speaking, however, the goal of a fully trained NN is for it to act successfully as a cloud parameterization step in a climate simulation, which could then in turn have significant benefits for biosphere-atmosphere interactions, as explained in Chapter 1.

CloudBrain

The question of whether SP can be represented as a nonlinear function can only be known by experiment, and the hyperparameters of a representative NN would require, as it does in many cases, a system of educated trial and error. It was possible that its processes could either not be captured in a known NN form, or that, if it could, suitable hyperparameters of the NN could not be found in a reasonable period of time.

It is reasonable to begin with a reduced complexity implementation of SP as a beginning point for training a NN as this is more likely to succeed than a more complicated version. Our NN was thus trained on an aquaplanet configuration of CESM. An aquaplanet is an idealized model configuration, with no land, topography, or ice; the planet is simulated as being completely covered with liquid water (Medeiros et al., 2016). Aquaplanet simulations have been used in the past to study the differences between resolved and parameterized models and other comparisons (Williamson, 2008; Blackburn et al., 2013; Williamson et al., 2013; Medeiros et al., 2015; Voigt and Shaw, 2015), so it is a reasonably good choice for both training and evaluating a NN-based CAM model, at least for internal atmospheric physics, before moving to the next step (Chapter 4) of examining consequent tradeoffs for biosphere-atmosphere interaction.

To further simplify the experiment, sea surface temperatures (SSTs) are homogenized in longitude such that each latitude has the same SST for every longitude, and the ocean does not interact dynamically with the atmosphere. Additionally, seasonality is removed. By seasonality, what is meant is that the simulated Earth's tilt and distance to the sun is fixed (in our case, at vernal equinox). Normally, changes in the Earth's tilt and distance to the sun cause parts of the planet to receive different amounts of solar radiation at different times of year, which influences the spatial and temporal intensity of energy fluxes and affects local and global climate, and generally complicates analysis of the simulated atmospheric dynamics.

For the training simulation, grid resolution of the global model is approximately two degrees (8192 grid cells) in the exterior, with 30 vertical layers; each grid cell of the global model has an embedded CRM with the same vertical grid and an 8-column subgrid spanning 32-km with 4-km (convection-permitting) horizontal resolution. The data used for training and validating the NN are the arterial inputs and outputs between the 8192 CRMs and their host global model. The CRMs and global model exchange information at every 30 minute global model timestep, and running the simulation for 2 years (after an initial 3 month spinup period) yields approximately 286 million samples (750 GB of data), half to be used for training and the other half for validation.

CloudBrain Software Development

When I was brought aboard, a prototype code base for this project, called Cloud-Brain (CBRAIN), was written in three programming languages. The model simulation code (CAM, SPCAM) was written in Fortran by teams of researchers over decades, coordinated by the National Center for Atmospheric Research in Boulder, CO. The data preprocessing code was written in MATLAB, and the code for training the NN was written in the Python. The machine learning Python package used for NN development and training was the Tensorflow package. Unlike CAM and SPCAM these were written by individual professors and research scientists at Columbia University and UCI, and a major part of my contribution to this Chapter is re-inventing them for improved usability, as described below.

Tensorflow is an open source machine learning system originally developed at Google that focuses on the creation, training, and evaluation of deep NNs. A deep NN is a NN with many layers, each layer typically having a large number of neurons. Tensorflow's unique draw is its relative ease of use in efficiently performing NN computations regardless of the hardware used, but is especially designed to maximally employ multicore CPUs, GPUs, and Google–specific hardware (Abadi et al., 2016).

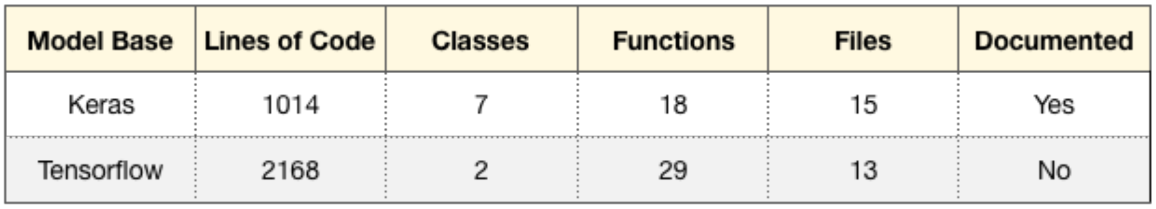
Tensorflow offers a great deal of customization in NN development. Many of the most common NN layer, neuron, and algorithm configurations are available out-of-the-box with little boiler-plate code, and most hyperparameter values are customizable. In the same vein, the amount of code needed for this level of customization is verbose, the learning curve can be steep, and poorly-documented user code on collaborative projects can cause a great deal of team confusion and headache.

The speed at which experiments can be performed is extremely important in research. Being able to quickly iterate and prototype ideas is a very fundamental scientific and software development practice, especially at the beginning of a new or creative project. Well documented code for key classes, functions, and algorithms is standard practice even for small groups of programmers in scientific environments (Baxter et al., 2006). Modular, easy-to-maintain, reusable code is the de-facto standard for object-oriented programming (Gamma et al., 2002).

As is common with many projects in which the value initial development speed prepends future usability, the Tensorflow code at the beginning of the project did not adhere well to standard design practices. The code base had no documentation, was not modular, and made it difficult to iterate very different prototypes from the current design. After all, it was written by senior researchers with limited free time to devote to this project.

Taking the advice of members of a machine learning-focused research group at UCI, I therefore rewrote the CBRAIN program, making heavy use of the Keras Python package. Keras is a Python package that acts as a higher level wrapper around Tensorflow (Chollet, 2015). In short, this meant that many fewer code lines were required for the same processes as pure Tensorflow code. The advantage of the Keras code I wrote is that it was easier to iterate prototypes, make modifications, and stay modular. Perhaps most importantly, the code base was well documented and extensible. When another team member joined the project, they were able to be brought quickly up to speed and extend the code base further.

After rewriting the code base, the number of lines of non-utility code was reduced by a full 50%, detailed in Table 3.1. This includes documentation added, so the actual number of executed lines is even more reduced than the table indicates, as the original Tensorflow code had no documentation. In addition to this large reduction, modularity was increased: the Keras code featured more files and classes despite having less absolute lines of code.



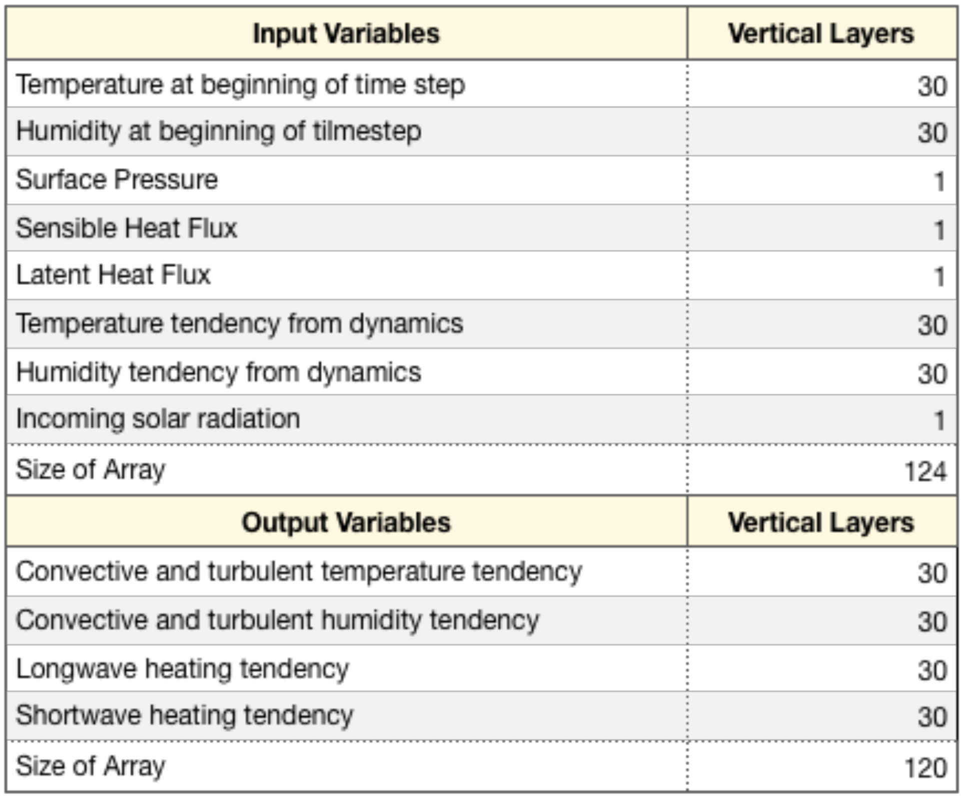
Model Base	Lines of Code	Classes	Functions	Files	Documented
Keras	1014	7	18	15	Yes
Tensorflow	2168	2	29	13	No

Table 3.1: A snapshot of code base characteristics at the stage in development when the Keras code base was first fully functional.

The big disadvantage of Keras over Tensorflow is that it sacrifices customization for simplicity. A drop in performance can be expected compared to results of a specific NN configuration achievable in Tensorflow using more fine-tuned parameters. However, in our case the ability to more quickly iterate on ideas and prototype new methods using Keras was more important than the ability to carefully fine-tune a model. Additionally, even though model accuracy initially decreased when we switched to Keras, as was expected, the final accuracy of CBRAIN written using Keras surpassed the accuracy of pure Tensorflow code.

Neural Network Specifications

During an SPCAM simulation, the physical subprocesses that SP predicts and modifies CAM after each step are the convective and turbulent temperature tendency, the convective and turbulent humidity tendency, the shortwave heating tendency, and the longwave heating tendency. This comprises the total heating and moistening profiles of the subgrids underlying the CAM simulation. The values for each variable are calculated for all 30 vertical layers in the simulation, so there are 120 values provided to CAM at each timestep for each grid cell. Naturally, those 120 values are the output of CBRAIN when given input. The input and output variables for CBRAIN are listed in Table 3.2. All variables were converted to the same units and normalized in a preprocessing phase. It was important to CBRAIN's skill that all inputs and outputs have a similar range of values, best between 0 and 1.



Input Variables	Vertical Layers
Temperature at beginning of time step	30
Humidity at beginning of tilmestep	30
Surface Pressure	1
Sensible Heat Flux	1
Latent Heat Flux	1
Temperature tendency from dynamics	30
Humidity tendency from dynamics	30
Incoming solar radiation	1
Size of Array	124
Output Variables	**Vertical Layers**
Convective and turbulent temperature tendency	30
Convective and turbulent humidity tendency	30
Longwave heating tendency	30
Shortwave heating tendency	30
Size of Array	120

Table 3.2: CBRAIN input and output variables.

A large variety of hyperparameter configurations for CBRAIN were investigated. The hyperparameter value space included:

(1) network depth (number of layers) and width (number of neurons per layer)

(2) amount of training data

(3) activation function

(4) optimizer (SGD algorithm used to update weights and biases) settings

(5) dropout value; normalization technique

CBRAIN's cost function was mean squared error (MSE). Figure 3.3a shows the sensitivity of MSE to number of network parameters and network depth. As can be seen, deeper networks performed better than shallower networks given the same number of parameters. The most accurate version of CBRAIN, which is the only one that will be discussed hereafter in Chapter 3, used 8 layers each with 512 neurons (see Figure 3.4). Dropout with a value of 0.5 was applied to each layer. Dropout is a technique where a fraction of the neurons and their connections in a layer are ignored during a training example, and has been shown to reduce overfitting (Srivastavaet al., 2014).

In Figure 3.3b, the sensitivity of MSE to amount of training data is shown. There is never such a thing as having too much training data, but in this experiment added-value drops off significantly after 3 months of data, and the gain in accuracy after 9 months of data is small.

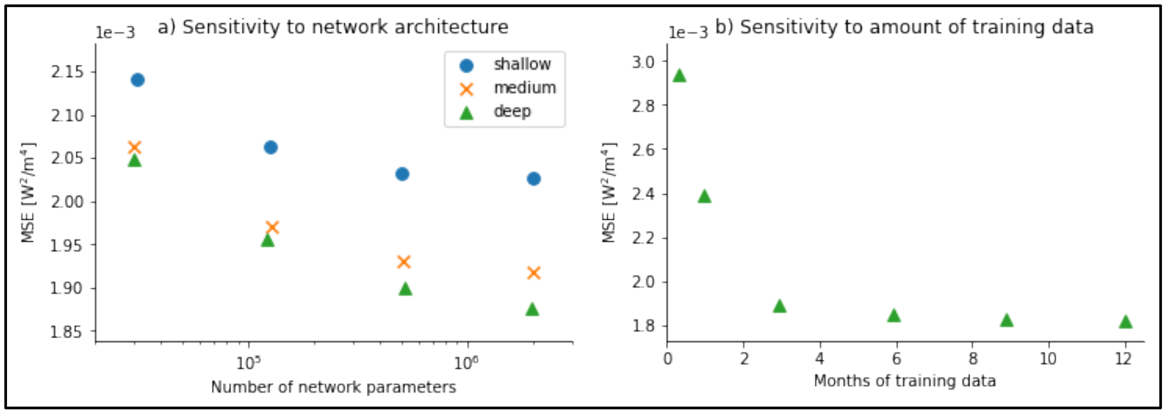


Figure 3.3: The shallow, medium, and deep networks in (a) had hidden layers of depth 1, 2, and 8, respectively, while maintaining the same number of parameters. (b) suggests that a year of very high-resolution training data could be sufficient for the training of a NN approximating SP.

The activation function used throughout CBRAIN, except in the final layer, is the Leaky Rectified Linear Unit (LeakyReLU), which is based on the Rectified Linear Unit (ReLU) activation function. ReLU is $max(0, x)$, and is popular for use in training NNs as it computes faster and empirically both optimizes more easily and converges more quickly than many sigmoid activation functions (Nair & Hinton, 2010; Zeiler et al., 2013). LeakyReLU, $max(\alpha x, x)$, is a variant of ReLU that has shown in cases to be more robust during training with suitable α than enforcing a hard-zero gradient with ReLU (Maas at al., 2013; Xu et al. 2015).

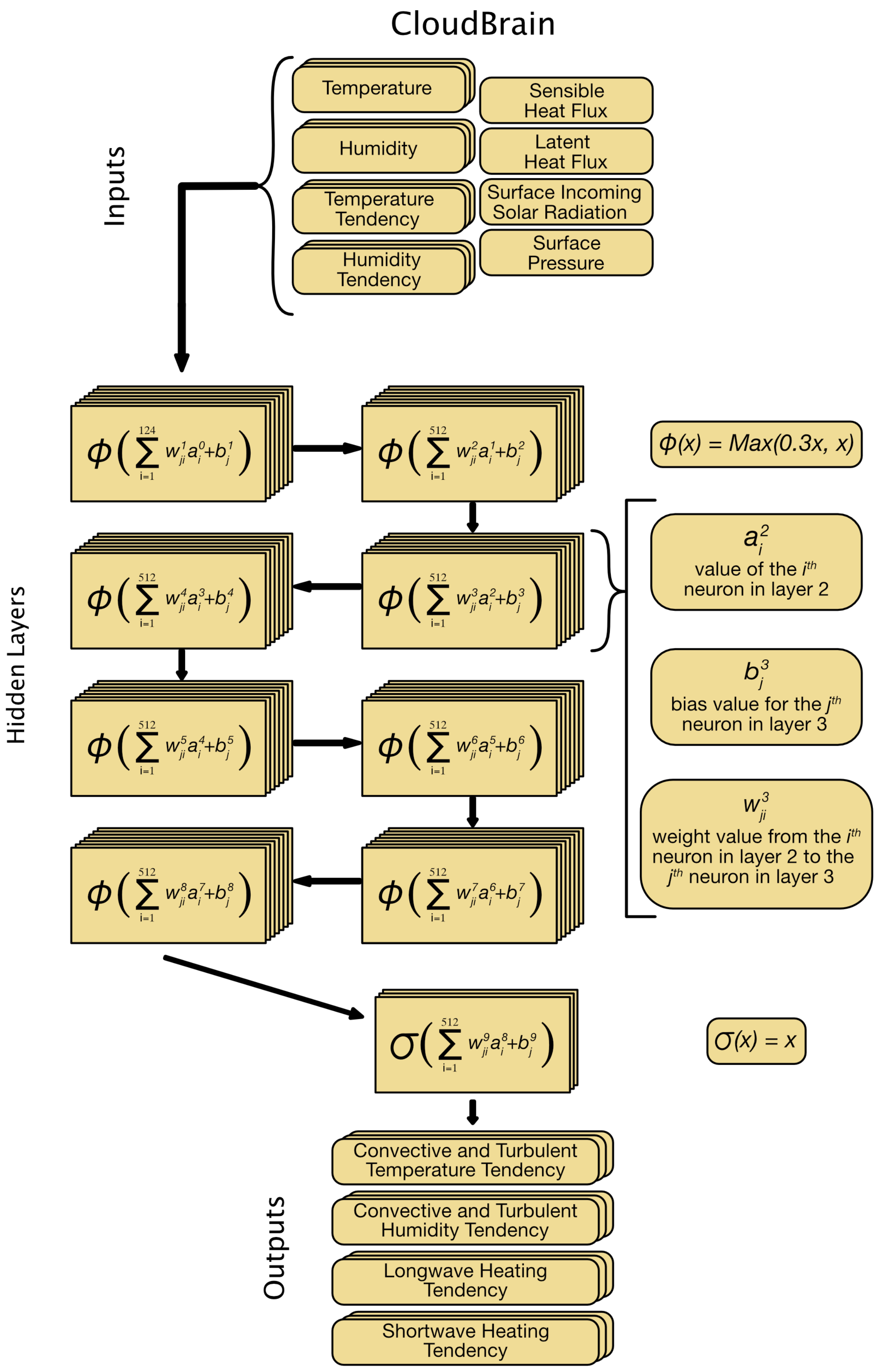
The minibatch SGD algorithm (optimizer) used during backpropagation for CBRAIN was Adam. Adam is short for “adaptive moment estimation”, and it computes adaptive learning rates for individual parameters, which has been shown to converge faster in many cases than regular GD (Sharma et al., 2017).

Adam works slightly differently from regular SGD in that each weight and bias has its own learning rate parameter, β. For a particular parameter k, the very first update Δk is based on the gradient of the loss with respect to that parameter:

$$\mathrm{k_t = \ k_t + \eta \Delta k_t}$$

However, at the next update step, Adam retains the “momentum” from the previous Δk, so the update becomes:

$$\mathrm{k_t = \ k_t + \eta \Delta k_t + \ \beta \Delta k_{t-1}}$$



CloudBrain
Inputs
Temperature
Humidity
Temperature Tendency
Humidity Tendency
Sensible Heat Flux
Latent Heat Flux
Surface Incoming Solar Radiation
Surface Pressure
Hidden Layers
$\phi\left(\sum_{i=1}^{124} w_{ji}^{1} a_i^{0} + b_j^{1}\right)$
$\phi\left(\sum_{i=1}^{512} w_{ji}^{2} a_i^{1} + b_j^{2}\right)$
$\phi\left(\sum_{i=1}^{512} w_{ji}^{3} a_i^{2} + b_j^{3}\right)$
$\phi\left(\sum_{i=1}^{512} w_{ji}^{4} a_i^{3} + b_j^{4}\right)$
$\phi\left(\sum_{i=1}^{512} w_{ji}^{5} a_i^{4} + b_j^{5}\right)$
$\phi\left(\sum_{i=1}^{512} w_{ji}^{6} a_i^{5} + b_j^{6}\right)$
$\phi\left(\sum_{i=1}^{512} w_{ji}^{7} a_i^{6} + b_j^{7}\right)$
$\phi\left(\sum_{i=1}^{512} w_{ji}^{8} a_i^{7} + b_j^{8}\right)$
$\phi(x) = Max(0.3x, x)$
a_i^2 value of the i^{th} neuron in layer 2
b_j^3 bias value for the j^{th} neuron in layer 3
w_{ji}^3 weight value from the i^{th} neuron in layer 2 to the j^{th} neuron in layer 3
$\sigma\left(\sum_{i=1}^{512} w_{ji}^{9} a_i^{8} + b_j^{9}\right)$
$\sigma(x) = x$
Outputs
Convective and Turbulent Temperature Tendency
Convective and Turbulent Humidity Tendency
Longwave Heating Tendency
Shortwave Heating Tendency

Figure 3.4: Visual representation of CloudBrain. For each training instance, there are 124 inputs values, 8 hidden layers with 512 neurons each, and 120 output values. Between the inputs and outputs there are 9 sets of weights and biases; each weight and bias is updated via the Adam optimizer at the end of each minibatch.

CBRAIN Results

CBRAIN predictions matched SPCAM behavior surprisingly well overall, particularly with heating rates as seen in Figure 3.5. Convective heating and moistening, radiative heating, shortwave absorption, and longwave cooling maxima are predicted at roughly the same horizontal and vertical locations (for more figures, see Gentine et al., 2018).

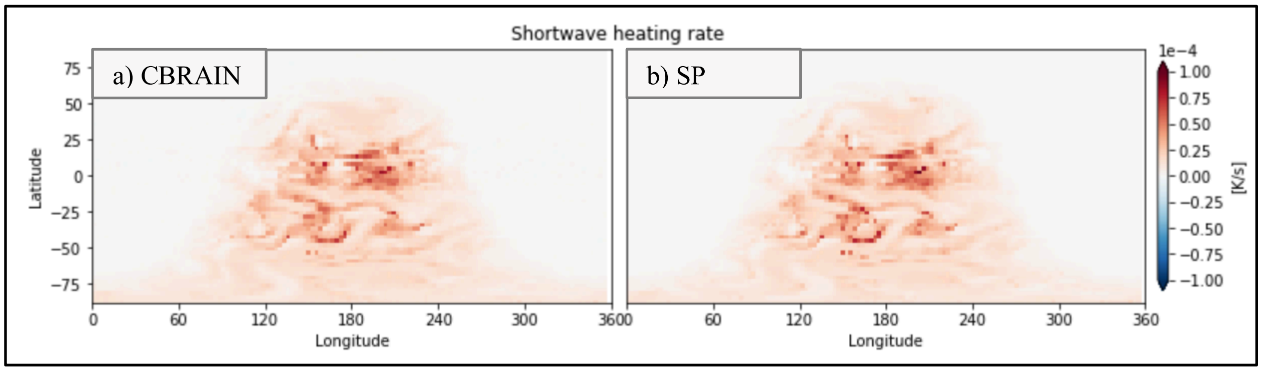


Figure 3.5: A slice in time of the shortwave heating rate for (a) CBRAIN prediction and (b) SP target for one sample by latitude and longitude for a mid-level vertical layer.

Figure 3.6 illustrates the R^2 statistics of CBRAIN by vertical layer averaged over latitude, longitude, and time dimensions. Longwave and shortwave heating rates have very high skill at most vertical layers, and convective heating and moistening rates also have similar high skill between 250 and 500 hPa. On the other hand, it can be seen that convective heating and convective moistening rates below the stratosphere perform worst at the boundary layer.

The reason for reduced skill at the boundary layer may be due to the fact that SP events at the boundary layers, including convective behavior, are much more stochastic than other layers, and CBRAIN has difficulty predicting stochastic behavior—this is common for neural networks, as they are, after all, deterministic approximations of nonlinear functions, and absent complex predictive ability produce values close to the mean state minimizes overall loss. This is a major challenge for processes in which stochasticity is important . CBRAIN's lower performance at the planetary boundary layer may have consequences for our over-arching interest of using this technique for improved biosphere-atmosphere interaction in climate models, which we will return to in Chapter 4.

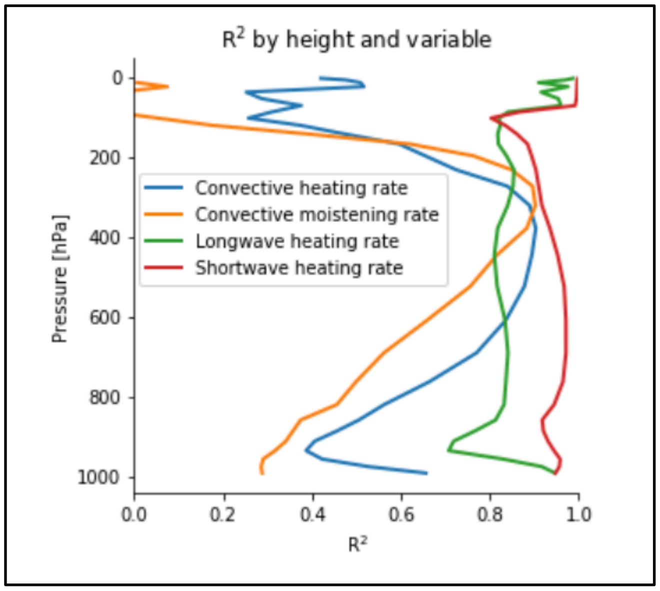


Figure 3.6: CBRAIN R^2 values for heating and moistening rates along the vertical column averaged over space and time. The y axis measures pressure; from a spatial standpoint 0 is at the stratosphere and 1000 is at the land-atmosphere boundary layer. R^2 is calculated as:

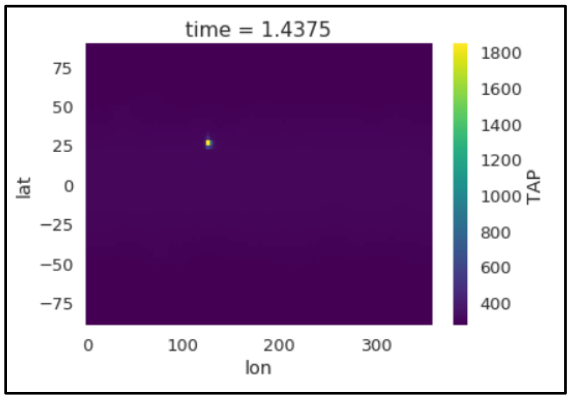
$1 - \frac{\text{squared error}}{\text{true variance}}$. The values are bound between 0 and 1. Scores closer to 1 signify greater accuracy.

Neural Network Community Atmosphere Model

Once trained, CBRAIN needed to be incorporated into CAM so that online simulations could be tested and analyzed. Beyond simply training a neural network, how skillfully a CAM simulation with CBRAIN representation of SP could reproduce SPCAM behavior is a crucial test.

Replacing the superparameterized CRM component of SPCAM with CBRAIN resulted in a new model that our team has begun to call NNCAM (Neural Network Community Atmosphere Model) in which the matrix biases and weights developed during our training procedure were further implemented as a forward prediction parameterization in the Fortran-based global climate model code.

CBRAIN does not inherently conserve energy, so a basic post-CBRAIN energy conservation step was implemented for NNCAM. Initial simulations, also called runs, did not make it past a few timesteps. A variable would increase exponentially and the run would crash. Figure 3.7a and 3.7b illustrate an example of this. A number of methods were employed to curb the runaway behavior of NNCAM, all related to the training and development of CBRAIN. We initially thought that certain weights in the network might have been becoming abnormally large and significant compared to other weights, and was causing the problem—upon examination however, there was no obvious culprit and we could not verify if any weight or series of weights were the source of the crashes. A more successful course of action was in normalizing the inputs by first calculating the mean and standard deviation of each variable; then subtracting the mean and dividing by the standard deviation; and finally dividing by the maximum value of its new range—previous normalization techniques sometimes resulted in division by very small numbers, which contributed to the point variables "blowing up" and crashing the run. A final step taken was to introduce a small number of stochastic samples into the training data for NNCAM to be more robust to atypical values. It did have the side effect of pushing NNCAM to become slightly less variable and more likely to predict values closer to the mean, but less likely to produce values that it will be unable to handle after a CAM timestep.

(a)


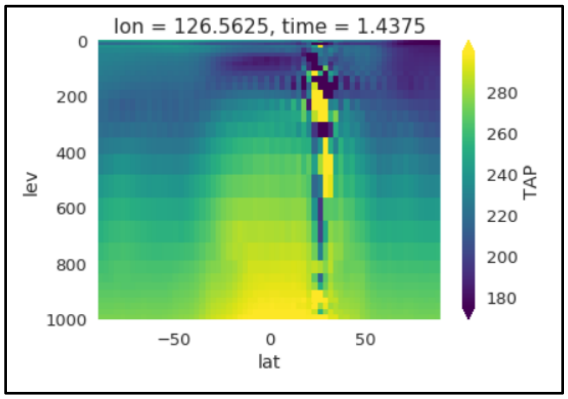
(b)


Figure 3.7: Example of a single grid cell increasing exponentially and crashing a NNCAM run view over (a) horizontal dimensions and (b) one horizontal and one vertical dimension. TAP is the acronym for "temperature after physics", in units of Kelvin. Methods to prevent this behavior are implemented in preprocessing and training, not during a simulation. Among others, this illustrates the importance of iterative and conscious design principles in overcoming software engineering obstacles.

After solving engineering challenges, the NNCAM simulations reproduced SPCAM behavior shockingly well. One surprise was that NNCAM exhibited energy conservation behavior before the explicitly enforced energy conservation step, and required less modification than expected, seen in Figure 3.8. This suggests that the new CBRAIN "learned" to enforce energy conservation through the data alone. Comparing NNCAM's speed with both CAM and SPCAM, NNCAM's parameterization step was 8 times faster than CAM's and 20 times faster than SPCAM's. Greater detail and analysis of NNCAM is presented in Rasp et al., 2018.

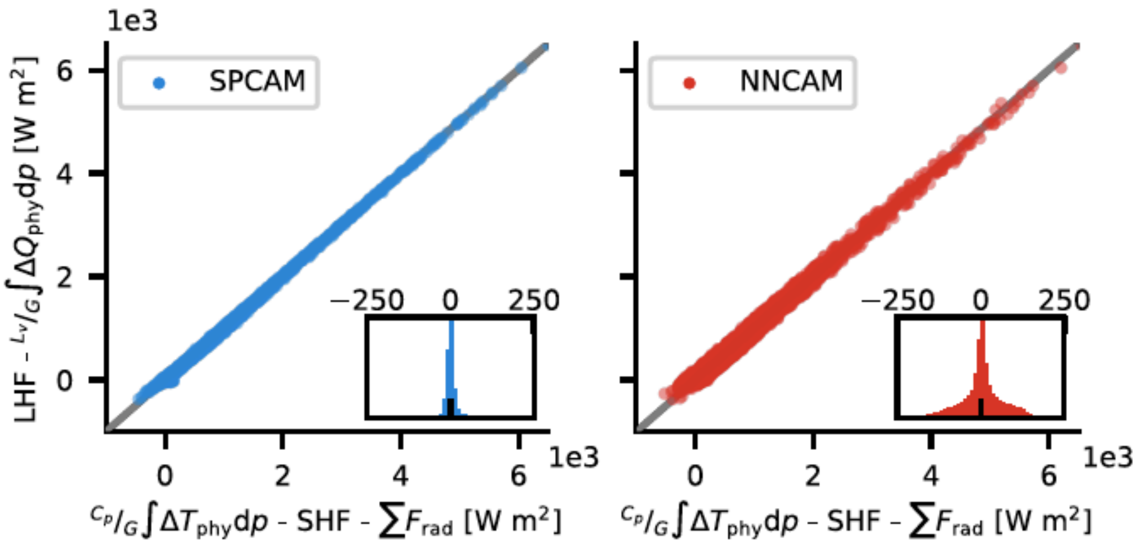


Figure 3.8: Column moist static energy conservation. Each point is a prediction at a single column, and the number of points corresponds to ten time steps. C_p is the specific heat of air, L_v is the latent heat of vaporization, and G is gravity. The *x* axis is calculated as the vertically integrated column heating ($C_p/G \int \Delta T_{phys} \mathrm{d}p$) minus sensible heat flux (SHF), minus the sum of boundary radiative fluxes ($\sum F_{rad}$). The *y* axis is calculated as latent heat flux (LHF) minus the vertically integrated column moistening ($L_v/G \int \Delta Q_{phys} \mathrm{d}p$). The grey

line *y(x) = x* represents perfect energy conservation. The insets are the distribution of differences between model predictions and perfect energy conservation.

Discussion

CBRAIN performed surprisingly well in emulating SP behavior, but looking to the future, there are a number of design features that could be made to develop a more sophisticated NN. CRMs run simulations alongside the GCM, and at every GCM timestep they exchange information. In CBRAIN's current design, all training examples are shuffled in time, so the time series dimension of the data is not taken into account during training. Skill could potentially be improved by implementing a recurrent neural network (RNN), which is a NN with cyclic paths that are designed to encapsulate patterns in sequential data (Lipton, 2015). Along the same lines, there is some information in the CRM that is not exposed to the GCM during the information exchange, such as mid-air liquid precipitation. Long short-term memory (LSTM) is a technique utilized in RNNs to retain information throughout the course of training with applicability for time-series prediction (Hochreiter & Schmidhuber, 1997). If the lack in stochasticity of the output of CBRAIN is determined to be severely detrimental to the skill of NNCAM, then designing CBRAIN as a generative adversarial network (GAN) could introduce a level of stochasticity (Goodfellow et al., 2014). The challenge of RNNs, LSTM, and GANs are that they are much more difficult to train than the simple deep neural network presented in this paper.

Despite its challenges, CBRAIN and NNCAM are great successes, and the formative work I performed during development helped lead to the Gentine et al., 2018 publication in *Geophysical Research Letters*, and pave the way for the Rasp et al., 2018 accepted paper in

Proceedings of the National Academy of Sciences. Analysis of NNCAM skill within the context of the biosphere is a critical next step, and is the focus of the work presented in Chapter 4.

CHAPTER 4

Land Response Effects to CBRAIN

Alongside cloud process parameterization issues, a major source of uncertainty in climate predictions is ecosystem feedbacks between the simulated biosphere and atmosphere that have impacts on global CO2 concentrations (Friedlingstein et al., 2014). SP has been shown to improve the representation of large-scale weather phenomenon such as the variability in El Niño-Southern and Pacific Decadal Oscillations (Krishnamurthy & Stan, 2015), and improves rainfall projections over the Amazon Basin (Zhang et al., 2017), which in a land model simulating carbon-nitrogen biogeochemistry has a profound impact on vegetation composition and carbon cycle fluxes (Chang et al., 2018). On the one hand, the idea of having a NN representation of SP is attractive for further improving the realism of the simulated land-atmosphere interface. For instance, retraining CBRAIN with a richer training dataset that is built on an SP augmented beyond its traditional limitations (i.e. with very high resolution CRMs capable of ever more realistic cloud-radiation transfer and precipitation physics) now seems like a viable possibility. Online incorporation of such a NN into a coupled land-atmosphere model could be a game changer for simulating complex vegetation-atmosphere interactions that depend sensitively on the turbulent processes that control planetary boundary layer height; these do not become explicitly resolved in SP until CRM grid resolutions approach 250-m horizontally and 20-m vertically, at extreme computational expense (Parishani et al. 2017). The idea is enticing since such a framework could significantly improve the realism of these land-atmosphere feedbacks in a next generation of ESMs.

On the other hand, there is reason to believe that success demonstrated for atmosphere-only aquaplanet simulations in Chapter 3 might not be replicated when land surfaces are included in the modeling framework. For instance, the worst performing vertical layers of CBRAIN in terms of skill at reproducing heating and moistening profiles were at the upper atmosphere and, more importantly to a land model, immediately atop the surface at the boundary layer. The lower skill at the land-atmosphere boundary layer, however, brings up an immediate question that can be tested. Namely, does the low prediction skill of our NN emulator of SP at the boundary layer translate into a corrupted representation of the land/forest response to SP? Or alternatively, are defects in the CBRAIN's ability to capture stochastic details of the planetary boundary layer immaterial to capturing the essential effects of SP on the simulated biosphere?

As a first step to address this issue, I conduct an idealized experiment to measure the similarity between a few key land-atmosphere response variables significant to the carbon cycle using CAM, SPCAM, and NNCAM atmospheric conditions.

NNCAM Configuration

The configuration for the parameterization step in NNCAM for the experiment has some differences from CBRAIN described in Chapter 3, but the aim remains to emulate SP behavior by using the atmospheric variables received from CAM at each timestep and vertical level to produce output variables that then dynamically influence CAM. Regardless of how well CBRAIN performs during training and testing, its feedback relationship with CAM during an actual run determines if its atmospheric simulation will emulate SPCAM, CAM, or is completely different from both. The number of variables present in the overall model make it such that even small differences in cloud parameterizations could lead to very different outcomes, which is the case for different parameterizations among ESMs and is the largest cause of spread in climate predictions (Flato et al., 2013).

The inputs and outputs for the parameterization step of NNCAM are listed in Table 4.1. The neural network used has 9 hidden layers of 256 neurons each, using the LeakyReLU activation function between each layer of the network except between the final hidden layer and the output layer, which has a linear activation function. Figure 4.1 shows the flow of the parameterization step in NNCAM.

Input Variables	**Vertical Layers**
Temperature	30
Humidity	30
Surface Pressure	1
Sensible Heat Flux	1
Latent Heat Flux	1
Meridonial Wind	30
Incoming solar radiation	1
Size of Array	94
Output Variables	**Vertical Layers**
Heating Rate	30
Moistening Rate	30
Precipitation	1
Shortwave flux at surface	1
Shortwave flux at top of	1
Longwave flux at surface	1
Longwave flux at top of	1
Size of Array	65

Table 4.1: Inputs and outputs of the neural network used for the parameterization step in NNCAM.

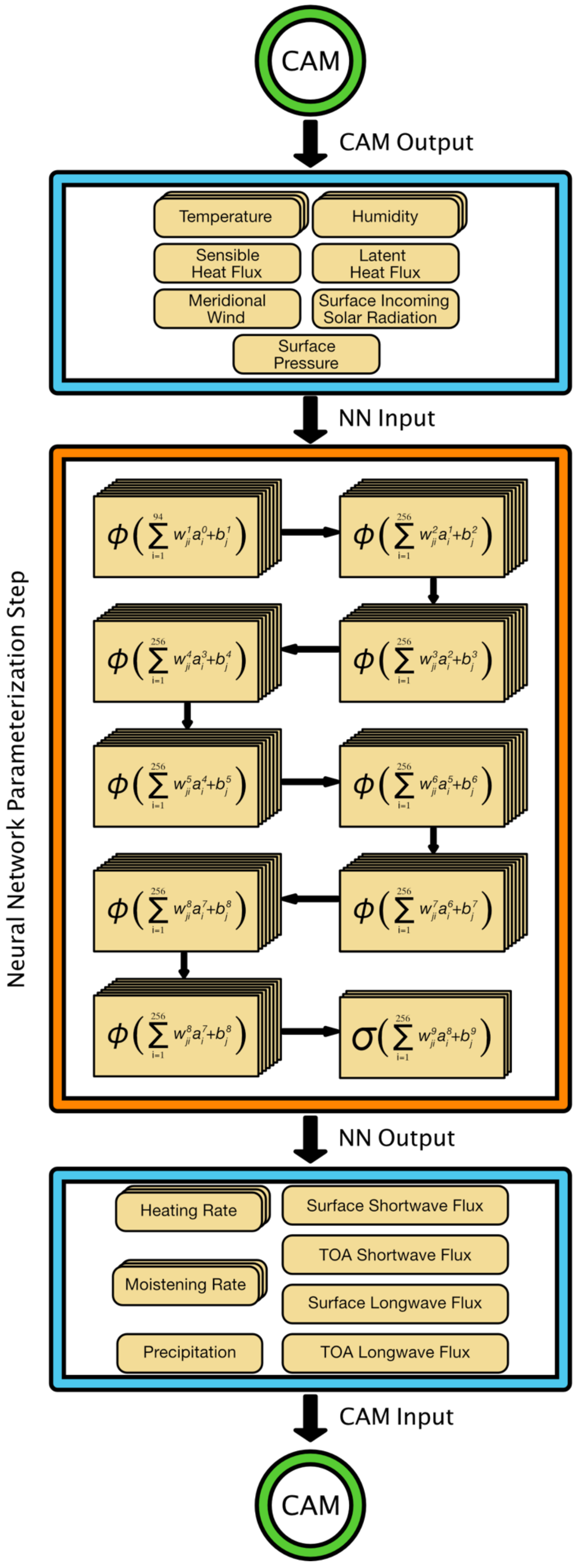

CAM
CAM Output
Temperature
Humidity
Sensible Heat Flux
Latent Heat Flux
Meridional Wind
Surface Incoming Solar Radiation
Surface Pressure
NN Input
Neural Network Parameterization Step
$\phi(\sum_{i=1}^{94} w_{ji}^{1}a_i^0+b_j^1)$
$\phi(\sum_{i=1}^{256} w_{ji}^{2}a_i^1+b_j^2)$
$\phi(\sum_{i=1}^{256} w_{ji}^{3}a_i^2+b_j^3)$
$\phi(\sum_{i=1}^{256} w_{ji}^{4}a_i^3+b_j^4)$
$\phi(\sum_{i=1}^{256} w_{ji}^{5}a_i^4+b_j^5)$
$\phi(\sum_{i=1}^{256} w_{ji}^{6}a_i^5+b_j^6)$
$\phi(\sum_{i=1}^{256} w_{ji}^{7}a_i^6+b_j^7)$
$\phi(\sum_{i=1}^{256} w_{ji}^{8}a_i^7+b_j^8)$
$\phi(\sum_{i=1}^{256} w_{ji}^{8}a_i^7+b_j^8)$
$\sigma(\sum_{i=1}^{256} w_{ji}^{9}a_i^8+b_j^9)$
NN Output
Heating Rate
Surface Shortwave Flux
TOA Shortwave Flux
Moistening Rate
Surface Longwave Flux
Precipitation
TOA Longwave Flux
CAM Input
CAM

Figure 4.1: Flow for parameterization step of NNCAM. The NN is trained on one year of SPCAM aquaplanet simulation data as in Chapter 3. The parameterization step takes place between CAM timesteps and produces values for the heating rates, moistening rates, precipitation, shortwave radiation flux, and longwave radiation flux. Each hidden layer has 256 neurons. ϕ is LeakyReLU, $f(x) = Max(0.3x, x)$, and σ is the linear function $f(x) = x$.

Community Land Model and the Amazon Basin

To test the difference in the land response when forced by NNCAM vs. SP-CAM, I compare 112-member ensembles of 5-year integrations of a single Community Land Model (CLM) land tile using one-way atmospheric forcing. The CLM version used is CLM v4, with active carbon–nitrogen biogeochemistry (CN) and constant CO2 and aerosol deposition levels from the year 2000. The CN component of the model defines each land cell to have pre-determined, static fractions of land types, the vegetative component of which has pre-determined, static fractions of plant functional types (PFTs).

Because atmospheric CO2 levels are static and the atmospheric variables are pre-defined, carbon-cycle and hydrologic-cycle feedbacks are disabled. Even though vegetative composition is unchanging, the model allows for the conversion of carbon into different forms within the land cell and release into the atmosphere. This includes processes such as leaf conversion to litterfall, either naturally (e.g. deciduous tree leaf fall) or through plant mortality. A fixed 2% of each PFT undergoes whole-plant mortality every simulation year, but they can also reach mortality through fire occurrence. Each PFT has a designated limit on temperature extremes it can endure and required amounts of nitrogen and soil water uptake. Root depth and resistance per PFT are taken into account for water uptake. If a fraction of a PFT dies on a land cell, that fraction is not able to be reestablished by another plant type, and it is able to regrow there again providing favorable conditions are present.

Soil moisture is prognosed on a 10-layer grid with unequally spaced depths; vertical soil moisture movement is determined by infiltration, diffusion, gravity, and root extraction. The water balance of the land tile is calculated as precipitation minus evapotranspiration, surface runoff, and deep soil drainage. Runoff and drainage from one tile do not affect adjacent tiles.

The land tile grid cell defining the soil moisture and vegetation properties of our terrestrial testbed was chosen based on real geography conditions at latitude -4.73 and longitude 295.0, in the Amazon Basin (see Figure 4.2).

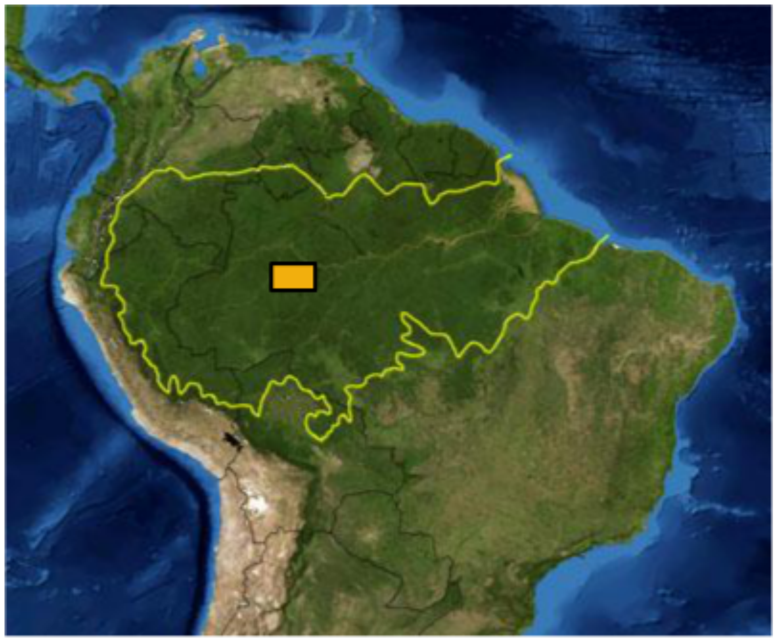


Figure 4.2: Land tile location in the Amazon Basin, a 1.9 x 2.5 degree land area centered on –4.73 latitude and 295.0 longitude.

Locating our testbed grid cell in the Amazon Basin was a strategic choice for several reasons. One of the greatest effects SP has on a climate simulation over traditional parameterization is that it more faithfully resolves tropical weather patterns and climate dynamics due to its treatment of convection and mesoscale processes, which become especially important near the equator (Randall et al., 2016). This includes its pattern of precipitation, which captures high rainfall events in the tropics (Kooperman et al., 2016b). Thus, aligning our atmospheric forcing input (which is taken from 112 separate grid points spanning 20S to 20N harvested from atmosphere-only aquaplanet runs for CAM, SPCAM, and NNCAM) and land model conditions to reside in the tropics is thus well situated to reveal consequences of these atmospheric effects on the simulated ecosystem. Further, a moist aquaplanet atmospheric forcing is more naturally suited to a grid cell in wet climate conditions than a grid cell in the middle of a desert. Finally, the tropics, and Amazon Basin specifically, are a prime location to test vegetative response variables because they have the most forest-type land cover in the world by both biomass and land area. Tropical forests sequester more carbon than any other land cover type, and the Amazon Basin itself has the most land area and above-ground biomass out of all tropical forests (Pan et al., 2011; Ahlström et al., 2017).

Some of the vegetative conditions of the land tile are important to point out. The land tile is 1.9x2.5 (latitude by longitude) degrees in size, and it is completely vegetative, with no river, lake, pasture, or urban fraction. About 96% is broadleaf deciduous tropical tree, about 4% is broadleaf evergreen tropical tree, and less than 0.1% is a combination of grass and bare soil. The dominant PFTs have characteristics that are important for the simulation in terms of survival. Both dominant tree types are assumed to have uniform canopy height of 13m; in more advanced versions of CLM the canopy is multi-layered, but CLM4.0 does not use a multi-layered canopy. The root depth of both tree types infiltrate below 8m, which is consistent with observations (Vogt et al., 1995). Both also do not have an upper limit on temperature for survival—put in other words, these two PFTs in particular will not die to heat stress at any upper temperature extreme. The Amazon is one of the most biodiverse ecosystems in the world (Malhi et al., 2008), but biodiversity is not captured in the model and more complex competition and ecosystem interactions are not taken into account.

Methods

One year's worth of atmospheric forcing from 112 atmospheric grid cells were selected from new 1-year long aquaplanet simulations for CAM, SPCAM, and NNCAM under the configuration listed earlier in Chapter 4. The atmospheric input grid cells were selected at 16 separate latitudes evenly spaced about the equator between -21 and 21 degrees, and 7 different longitudes roughly evenly spaced around the globe. In this way a set of forcing samples representative of a large near-equatorial geographic band was chosen with atmospheric inputs spanning a range of precipitation, temperature, and solar radiation distributions representing the tropical mean environment. The intent is to observe what a roughly global mean in land response to representative variations of atmospheric forcing would look like.

The atmospheric forcing variables from the grid cells were used as the inputs for a 112-member ensemble of single grid cell CLM integrations. A CLM atmospheric forcing simulation requires that downwelling surface solar radiation and precipitation be prescribed every six hours; and surface pressure, specific humidity, surface temperature, and wind magnitude be prescribed every three hours.

The CLM simulations were 5 years in length in order to expose not only the fast response of the land to swapping the essential nature of its atmospheric inputs, but also the slow drift to independent attractors, which provides a further test of emulation skill. The 1-year length of atmospheric forcing conditions was repeated for each of the 5 years; in other words, atmospheric conditions were identical from year to year for all 5 years. The land tile in every simulation had identical boundary conditions. 112 CLM simulations were run for each atmospheric model (CAM, SPCAM, and NNCAM), for a total of 336 CLM integrations. The results from each model were averaged together and the standard error across the ensemble calculated for each model to discriminate detectable differences between models from representative geographic variability throughout the tropics. Additionally, a single fully coupled 5-year CAM-CLM simulation was performed for comparison purposes, and as a benchmark calculation that avoids our idealizations by using realistic geography, full seasonality, and interactive atmosphere-land feedbacks globally. The ocean and ice conditions of the global model were prescribed in the coupled simulation.

A limitation of this work is that it focuses exclusively on testing the sensitivity of the land model to varying sets of atmospheric forcing patterns. Testing whether the NN-forced land response is similar to the SP-forced response is a logical first step towards answering the broader question of whether NN representations of atmospheric physics can do justice to fully interactive land-atmosphere feedback dynamics. The answer is not obvious since the NNCAM land-atmosphere boundary layer is both less skillful and less stochastic than SPCAM, such that it is worth exploring if non-coupled simulations produce similar land responses. For instance, if they do not, then the focus should be on improving the skill of NNCAM at the boundary layer before attempting to create a coupled version. Additionally, if the land response is similar in a one-way atmospheric forcing case, but in the future coupled cases differ significantly, then it would be strong evidence supporting that unpredictable aspects of turbulence are critical to land-atmosphere feedback dynamics (NNCAM has an inherently smoother boundary layer due to CBRAIN's training process, where in areas of low skill moving towards a mean state is the most effective path to reducing the loss of the cost function).

The one other limitation that must be first discussed relates to the experimental design in that different ensembles of atmospheric forcings are tested, but only a single land grid cell at fixed initial conditions is used. Would land grid cells with different properties or different initial conditions give much different results? While this is reasonable to expect, the author argues that carefully selecting an appropriate land tile and closely examining it under an ensemble of atmospheric conditions is more useful for the purpose of comparing CLM response to NNCAM and SPCAM than selecting many different tiles. Choosing multiple tiles would obscure details of CLM response sensitive to land type and initial conditions, and limiting the attention to a single land tile is a logical choice to avoid averaging out the response of CLM to different atmospheric forcings at the same time that it is critical to focus on those responses. To compare land responses to atmospheric forcings from idealized aquaplanet simulations, isolating variables by maintaining identical initial conditions and land properties is essential.

These limitations acknowledged, there are multiple reasons to motivate the null hypothesis that for land responses NNCAM will not be able to emulate the consequences of

SP for the simulated biosphere, as a number of potential problems can be anticipated. Specifically, these are our null hypotheses:

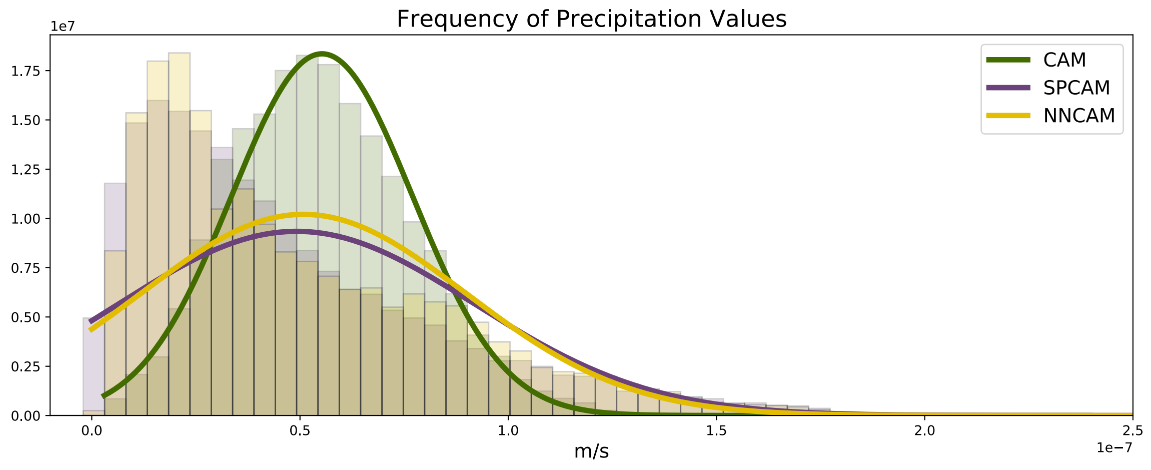
(Hypothesis 1) CBRAIN was trained on SP under aquaplanet conditions, which has no land and is not be suitable to use as atmospheric forcing. If the land response to SP turns out to be similar to that of a coupled run, this issue is not pertinent.

(Hypothesis 2) Without the full feedback effects of a coupled land-atmosphere model, the difference among forcings will be not be significant enough by themselves to produce significant differences in biogeochemical land response variables in a 5 year period. That is, the effects of SP on the land in our setup may be too subtle to detect even in benchmark tests, let alone the fidelity of a NN emulation of SP.

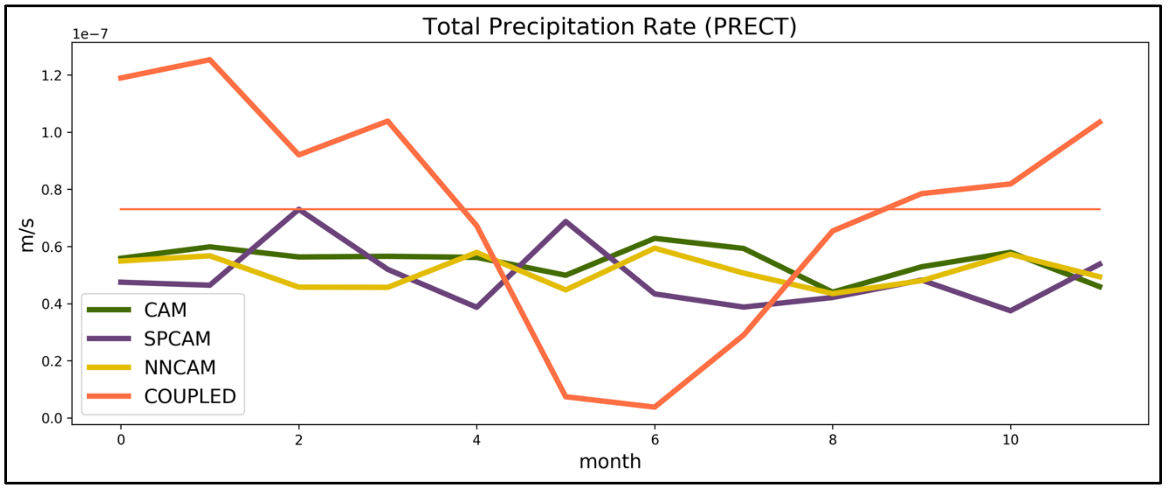
(Hypothesis 3) CLM will be crash when using NNCAM atmospheric forcing values. Occasionally some variables of the NNCAM should be expected to take on unreasonable values that might break the land model due to the noise inherent in NN predictions. For example, there will be slightly negative downwelling solar radiation, or surface pressure will be 0 (both of which are physically impossible). CLM has carbon, water, and energy balance check steps that, if they fail, force the simulation to stop. It is not a given that the land model will even be able to run without crashing when given NNCAM atmospheric forcing, since land models contain many more degrees of freedom and complexity than exist in the simpler aquaplanet testbed analyzed in Chapter 3; in the forthcoming tests, there are more potential pathways to a failure mode.

Certain differences in the resulting CLM responses can be expected based on speculation about the differences in the atmospheric inputs alone; indeed the effects of SP on terrestrial dynamics are only partially explored in the literature (Qin et al., 2018; Sun &

Pritchard, 2016). Effects of SP on precipitation have been noted by many (Kooperman et al., 2016b; Kooperman et al., 2016a). Taking a look at Figure 4.3a, NNCAM and SPCAM both capture precipitation extremes, but as can been seen in both Figure 4.3a and Figure 4.3b, the mean precipitation is lower than in CAM. Although the mean precipitation is slightly lower in SPCAM (and NNCAM) than CAM, this effect is so subtle compared to the effects of SPCAM on incoming shortwave radiation, that we suspect the latter will dominate overall consequences for carbon sequestration. As an aside, it is worth noting that the seasonally invariant precipitation rate of the idealized aquaplanet simulations lies between the seasonal extremes of the coupled run—this is a reassuring indicator that the atmospheric forcing of the aquaplanet simulations do not have completely unrealistic values for the selected land tile.



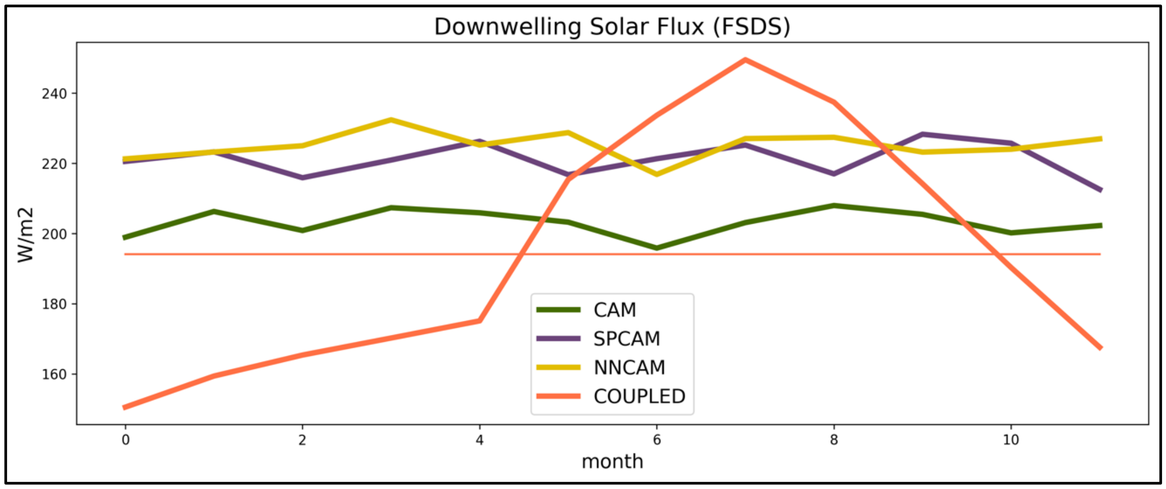
(a)



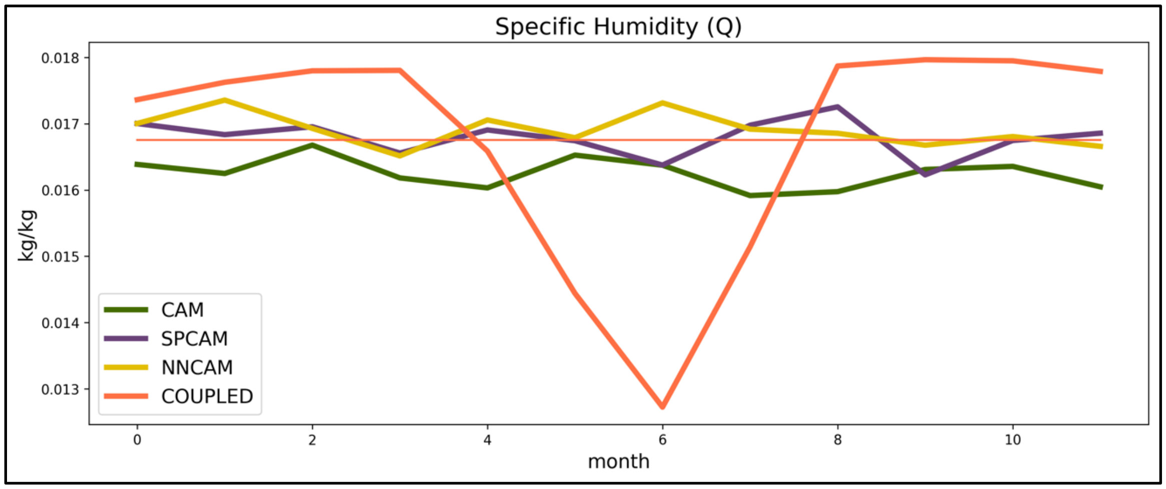
(b)

Figure 4.3: (a) is the frequency of precipitation among all atmospheric inputs for each model. Combined, NNCAM and SPCAM have a greater frequency of high-precipitation events than CAM, but the mean is lower. (b) shows the monthly mean rainfall of the ensembles for each model, including the average for each monthly mean of the 5 year coupled simulation. The coupled simulation exhibits the seasonal behavior of the Amazon, with wet and dry seasons. The precipitation of the aquaplanet simulations are seasonally invariant, by design. The thin line is the mean precipitation overall for the coupled simulation.

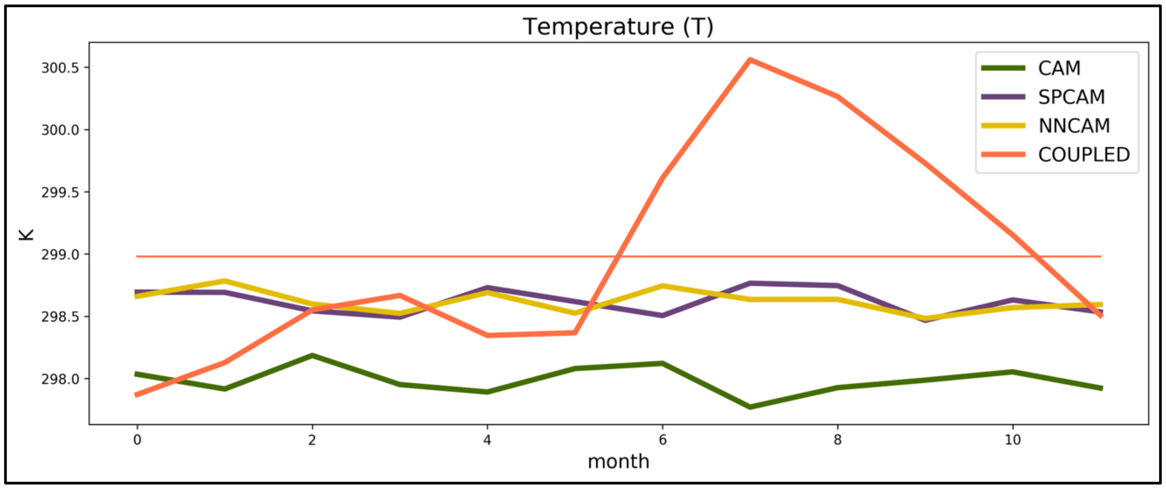
That NNCAM's precipitation pattern so closely matches SPCAM comes a surprise given CBRAIN's performance metrics in Figure 3.6 and characteristic low boundary layer variability, as in this configuration precipitation is directly predicted during NNCAM's parameterization step. However, NNCAM's other atmospheric input means and variances are also within reasonable range of SPCAM. Time series of the monthly means for CAM, SPCAM, NNCAM, and the coupled simulation are shown in Figue 4.4, and suggest that, depending on its sensitivity, CLM land response might be expected to be very similar between NNCAM and SPCAM, although the complex CN interactions of the CLM make it impossible to know without empirical tests.



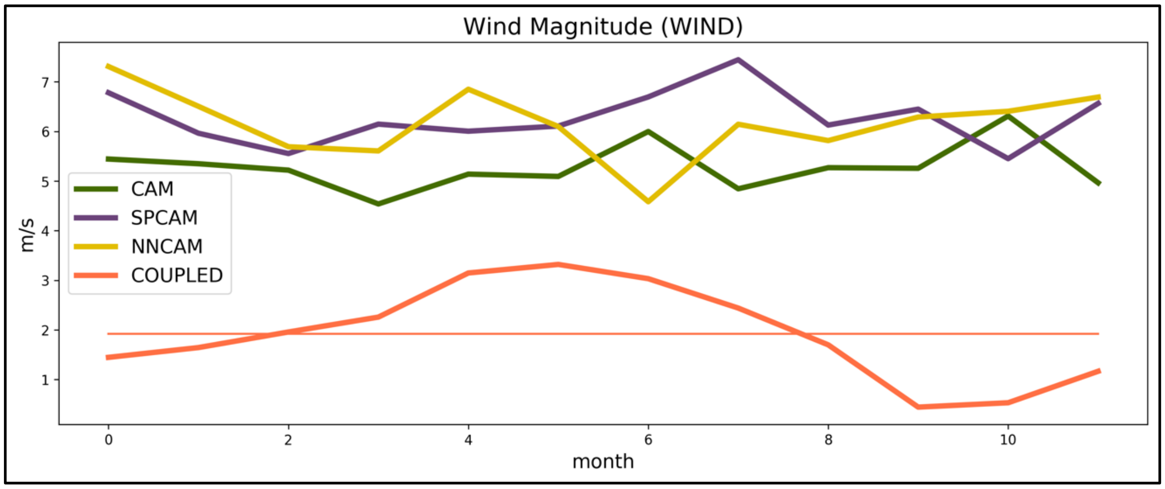
(a)



(b)



(c)



(d)

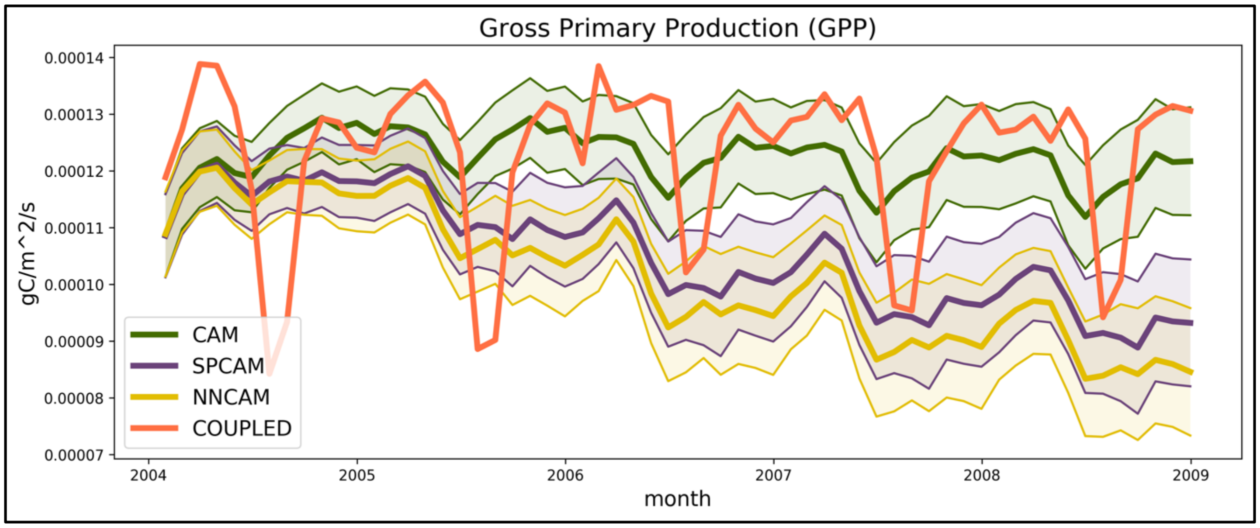
Figure 4.4: Monthly means of (a) downwelling solar flux, (b) near-surface specific humidity, (c) near-surface temperature, and (d) wind magnitude for each model ensemble, plus the average of the monthly means of the 5-year coupled CAM simulation. The thin line in each figure is the overall mean of the benchmark coupled simulation. The surface wind speeds in the aquaplanet simulations are fast due to the absence of surface friction from mountains and vegetation.

Looking at Figure 4.4a and 4.4b, the solar flux and humidity values for the aquaplanet simulations are within the upper and lower bounds of the coupled simulation, and in Figure 4.4c temperature values for the aquaplanet simulations are representative of the cooler wet season of the Amazon, which are more promising signs that the atmospheric forcing simulations do not have unreasonable values for the selected land cell. The wind values in 4.4d, however, are markedly different. This is to be expected due to differences in land surface versus ocean surface. Given that the temperatures of the models are not in any kind of extremes, and heat stress does not have an effect on the PFTs of the model, we could expect the higher mean temperatures of SPCAM and NNCAM to have no negative impact on the vegetation. In a coupled simulation, feedback effects could be expected from higher temperatures, but with atmospheric forcing this would not be the case. The wind speed values could have a large effect on soil evaporation. However, evapotranspiration as a whole will likely be dominated by transpiration (Chang et al., 2018), which is limited by the raw photosynthetic activity of the PFTs, in turn constrained by available net radiation and subsurface soil moisture.

Based on these atmospheric inputs, it is logical to predict that the gross primary productivity (GPP) of SPCAM and NNCAM should be systematically higher than CAM purely due to its much higher downwelling solar flux, which determines the amount of light available for photosynthesis. Concurrently, net ecosystem exchange (NEE) for SPCAM and NNCAM would be a stronger carbon sink than CAM. While precipitation does impact photosynthetic activity, the lack of feedback in the experiment would lead one to expect that the differences in precipitation among the models appear subtle enough that they would not differentiate the GPP or NEE.

Results and Discussion: CAM vs SPCAM vs NNCAM

Figures 4.5a and 4.5b display the 5 year time series of the monthly mean GPP and NEE of the CLM response to the three atmospheric forcing ensembles, with reference to the coupled simulation. The ensemble mean and standard errors are indicated with lines and shading. The first and most important result to note is that beginning at about a year and a half into the simulations, the SPCAM and NNCAM tropical mean GPP begins to diverge detectably to a lower equilibrium than simulated by CAM forcing. By the end of the 5 year simulation, there is 95% confidence that GPP and NEE for SPCAM and NNCAM are lower than CAM. The null hypothesis is rejected for statistically insignificant difference between SPCAM and CAM, and NNCAM and CAM. This supports the alternative hypothesis that the land response to NNCAM forcing is statistically similar to that of the SPCAM forcing. Moreover, even the unsteady details (rate of change and intraseasonal fluctuations) of the slow drift to the SP-forced GPP attractor are captured by the NN emulator of SP. Hypotheses 2 and 3 outlined at the beginning of this chapter must be rejected, as the land response, and in particular the ecosystem response, to SPCAM and NNCAM maintain the similarity carried over from the atmospheric simulations. The skill of NNCAM is maintained despite the relative poor fit noted for details of the boundary layer by Gentine et al. 2018, and the land model ran successfully even under the noise-prone NNCAM forcing, i.e. passing the model's carbon, water, and energy flux checks despite no preprocessing and the occurrence of unrealistic values at a percentage of its timesteps, particularly solar radiation. For example, examining the NNCAM atmospheric forcing ensemble mean at each time step reveals that solar radiation is slightly negative 13% of the time (with a minimum of -2.3 W/m^2), but it is not significant enough at any particular instance to fail the model's energy checks.



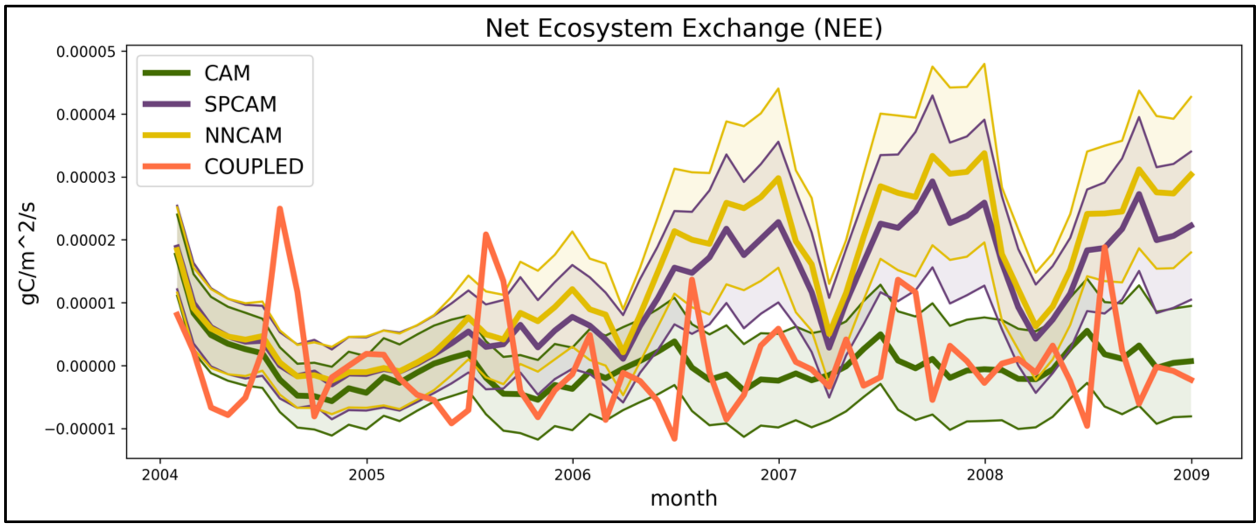
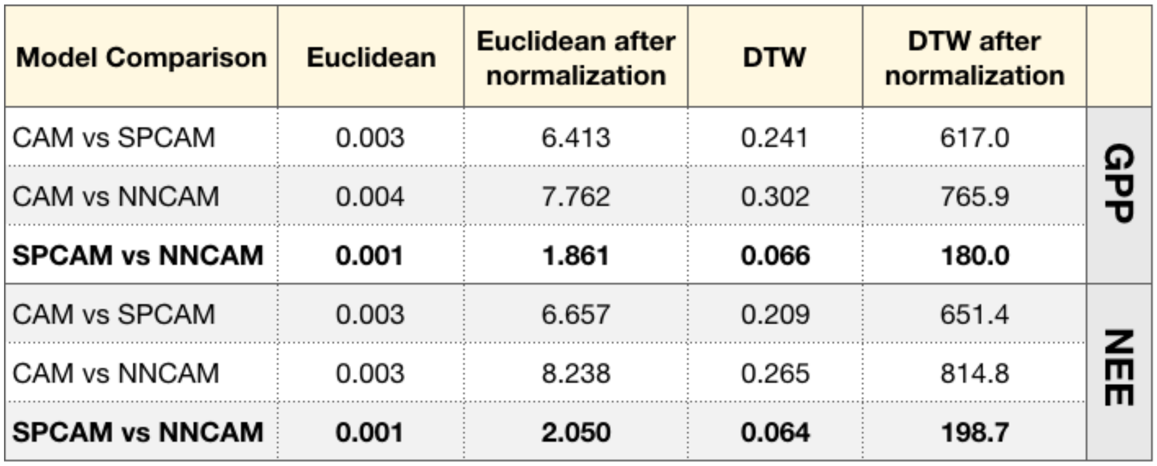


Figure 4.5: (a) GPP and (b) NEE monthly means for the ensemble of 5 year simulations. The thicker lines are the mean values, and the thinner surrounding lines with fill are the monthly mean standard error at each timestep—that is, the standard error was calculated for every timestep and averaged together monthly, instead of calculating the standard error of the monthly means themselves. The CAM simulation has remarkably similar values to the coupled simulation, and SPCAM and NNCAM are not in unreasonable ranges, so Hypothesis 1 from earlier in Chapter 4 must be rejected.

It is visually apparent in Figure 4.5, but measuring the differences in time series through standard techniques is an important task for data concerning ecosystem dynamics (Lhermitte et al., 2011). Appendix A details numerical tests measuring whether the time series of NNCAM and SPCAM are more similar to each other than they are to CAM on both a timestep basis and pattern-matching basis. The results of the time series comparison for NEE and GPP among the models are shown in Table 4.1. No matter which technique is used, SPCAM and NNCAM are more similar to each other than to CAM, as expected based on Figure 4.5.



Model Comparison	Euclidean	Euclidean after normalization	DTW	DTW after normalization	
CAM vs SPCAM	0.003	6.413	0.241	617.0	GPP
CAM vs NNCAM	0.004	7.762	0.302	765.9	
SPCAM vs NNCAM	**0.001**	**1.861**	**0.066**	**180.0**	
CAM vs SPCAM	0.003	6.657	0.209	651.4	NEE
CAM vs NNCAM	0.003	8.238	0.265	814.8	
SPCAM vs NNCAM	**0.001**	**2.050**	**0.064**	**198.7**	

Table 4.2: RMSE and dynamic time warping among models for GPP and NEE before and after normalization. Lower scores indicate greater similarity. As can be seen, SPCAM and NNCAM are more similar to each other than to CAM regardless of the time series measurement used. For more detail on the calculation of these values, see Appendix A.

A look at the data is warranted to determine why NEE increases (GPP decreases) for SPCAM and NNCAM, relative to CAM. This result is contrary to our original expectation based on the working hypothesis that the effects of SP on solar flux would be the greatest factor. Given that the cause must be due to a difference in atmospheric forcing, the atmospheric variable that is a likely candidate must be one that effects slow changing variables or has compound effects over time, such as temperature and precipitation (despite precipitation's apparently subtle drop relative to CAM noted earlier). Figure 4.6 shows the time series for each model for photosynthesis. The mean for SPCAM and NNCAM are lower after half a year, but at a year and a half they are markedly lower. Photosynthesis is a primary driver of GPP, and so the cause in GPP difference among models can be attributed to the difference seen in the time series of photosynthesis.

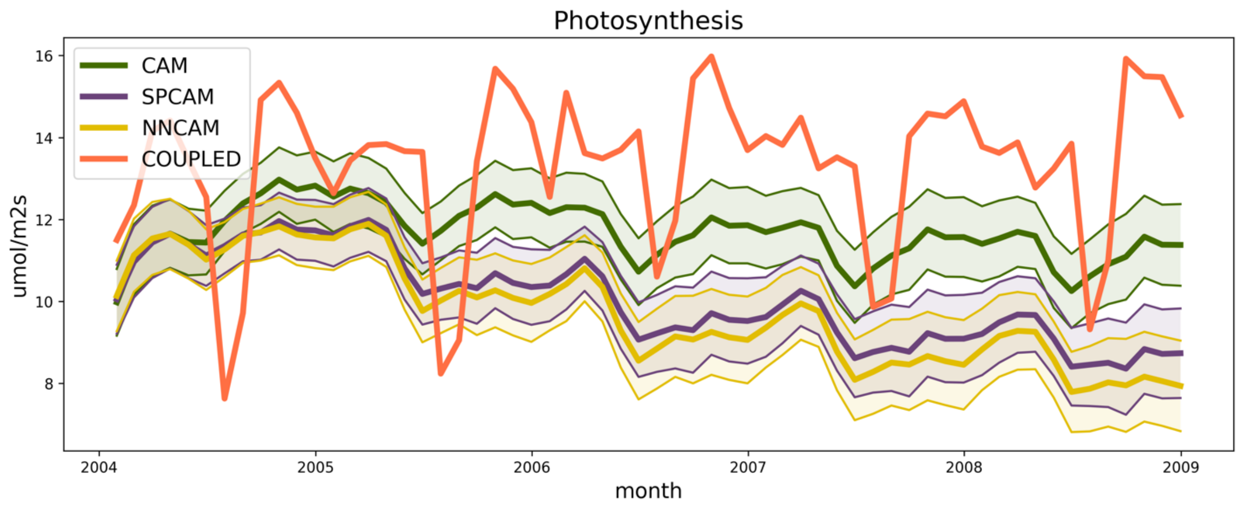


Figure 4.6: Time series for photosynthesis. Thicker lines indicate mean values, and the thinner surrounding lines with fill of the same color are the monthly mean standard error at each timestep.

To better illustrate some of the complexities of model interactions, Figure 4.7 simplifies some of the complex biogeochemical relationships that are important to GPP and NEE within CLM. Provided there are no limiting nutrients, soil moisture along with solar radiation (not shown) drives photosynthesis and GPP. The more photosynthesis occurs, the more transpiration exists, which cools down the land and vegetation as latent heat is carried into the atmosphere. The overall lower mean precipitation of SPCAM and NNCAM may be the cause of lower amounts of GPP over time (hereafter I will only refer to SPCAM, as the relationship between SPCAM and CAM has been shown in this chapter to be statistically similar to that of NNCAM and CAM). Tropical trees in the Amazon have deep roots that extend over 8m into the soil; these roots are used to access deep soil moisture during the dry season (Nepstad et al., 1994; Huete et al., 2006). A critical breakpoint in precipitation has been observed in previous Amazonian model simulations that is required to recharge deep soil moisture (da Costa et al., 2010; Baker et al., 2009; Ahlström et al., 2017). If not enough rainfall occurs during the wet season the trees run out of water and can have high rates of mortality (Allen et al., 2010). It should be noted, however, that changes in Amazon rainfall differ among models (Li et al., 2006; Powell et al., 2013), and observationally the Amazon is more robust to drought than models predict (Saleska et al., 2007).

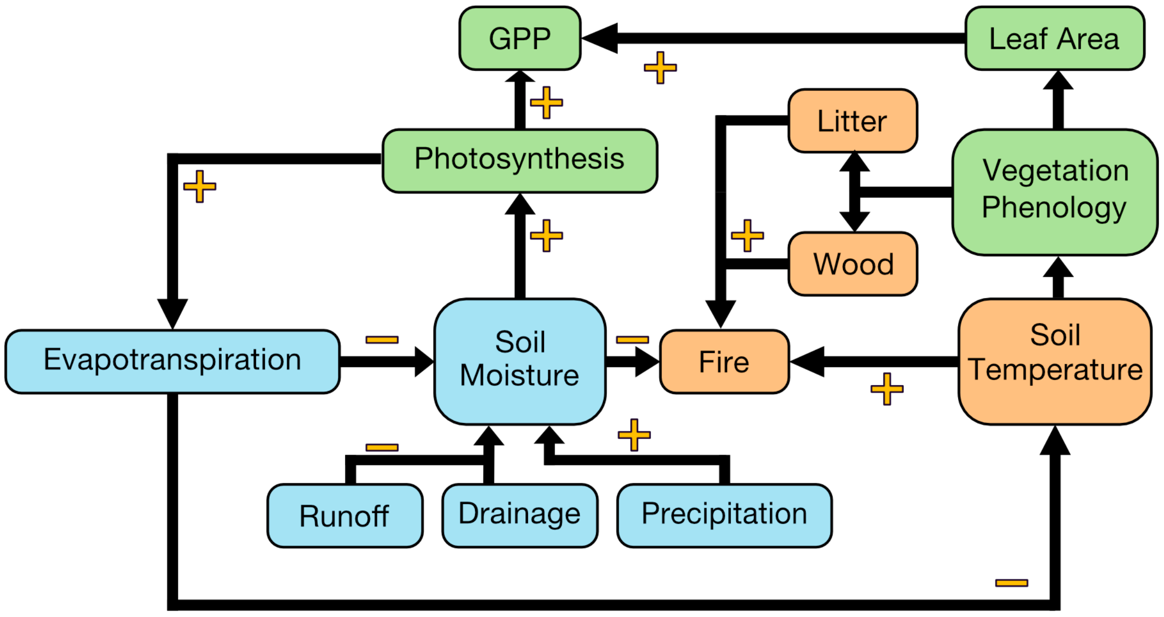
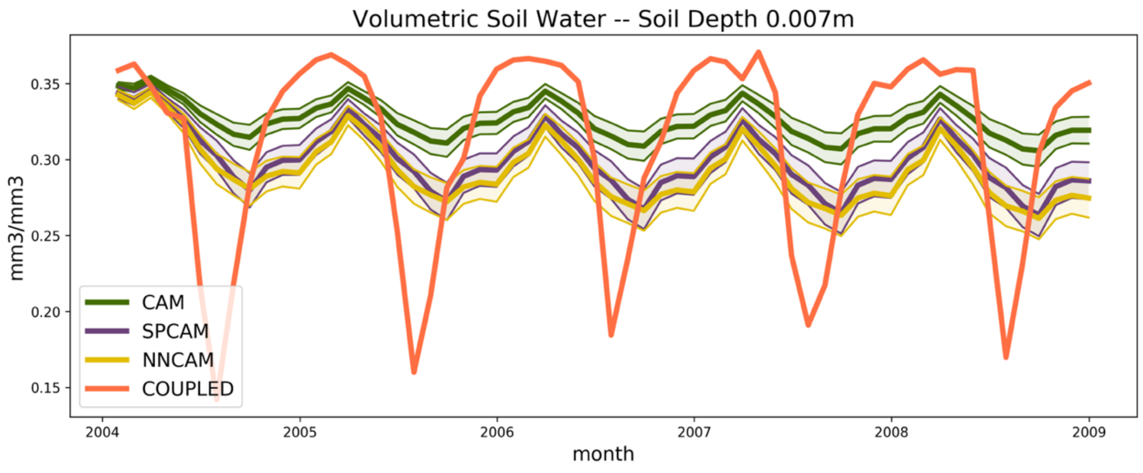
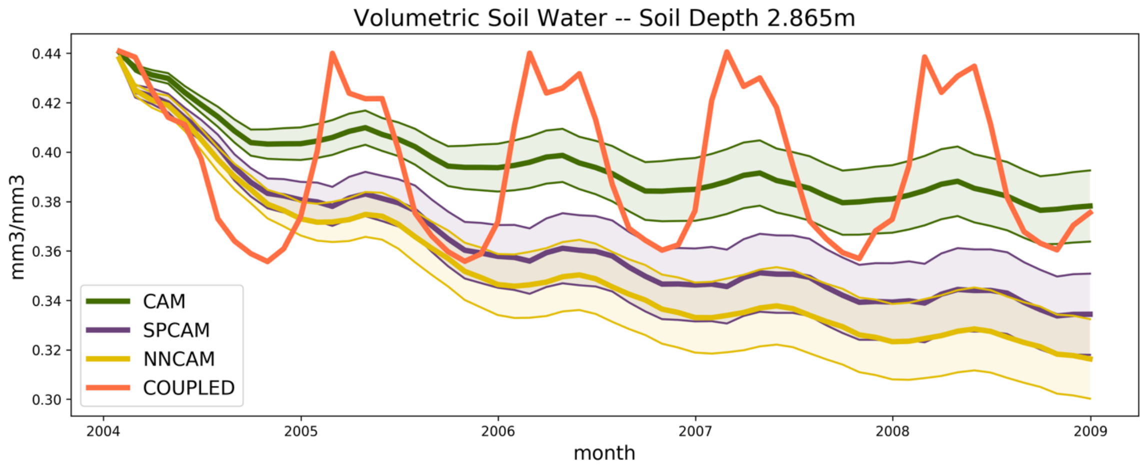


Figure 4.7: An overview of part of the complex interactions among variables in CLM. The plus signs indicate a positive relationship occurring in the direction of the arrows, the minus signs indicate an inverse relationship in the direction of the arrows, and arrows without a plus or minus sign indicate that there can be varying effects. The blue, orange, and green variable colors group them roughly into moisture, fire, and vegetation categories. Soil moisture, and thus precipitation, is a major component in this graph, as it has direct and indirect impacts on all other variables.

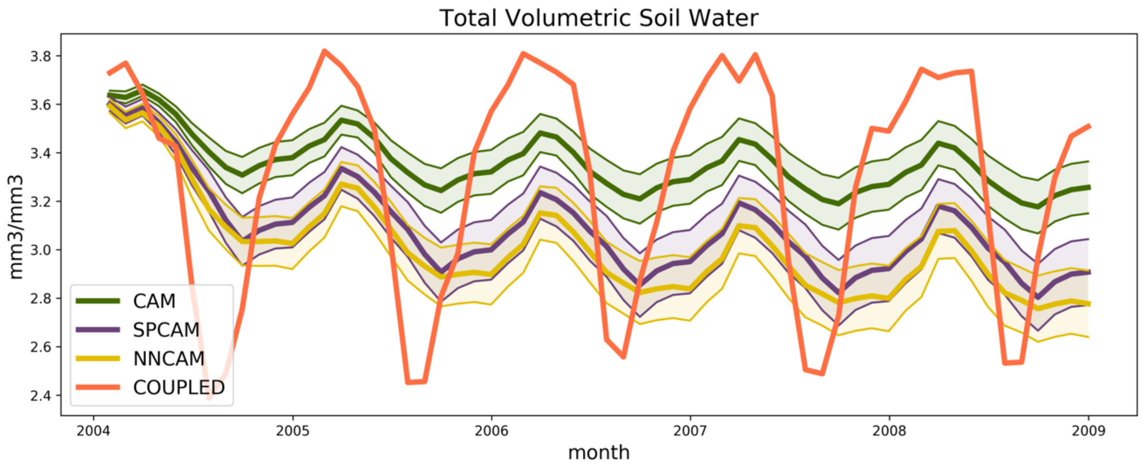
If the deep soil moisture of SPCAM-forced CLM simulations slowly depletes but CAM's does not, then soil moisture limitations may have acted to lower photosynthesis, and thus GPP. This is confirmed by analysis of the volumetric soil water concentrations along increasing soil depths in Figure 4.8a and Figure4.8b. Surface soil concentrations vary little from year to year, but the water concentration at lower depths drift very slowly to systematically drier conditions under our idealized SPCAM forcing. By the end of year 2, deep soil moisture declines to even lower levels than the coupled simulation's seasonal minimum, which supports the hypothesis that deep soil moisture may be a limitation on photosynthesis. However, 4.8c reveals that overall the difference in soil moisture between SPCAM and CAM remains relatively constant from year to year, which might indicate that simply less soil moisture overall, and not necessarily deep soil moisture, is the cause of lower GPP. The variance between CAM and SPCAM precipitation throughout the simulation is greater than that of the combined effects of evapotranspiration (ET), runoff, and drainage, shown in Figure 4.8d. Nothing in the data indicates that the combined ET, runoff, and drainage rate is relative to the level of mean precipitation differently between SPCAM and CAM, and so provided the soil moisture is the limiting factor on photosynthesis, the lower mean precipitation itself is the driving factor for the difference seen in GPP.



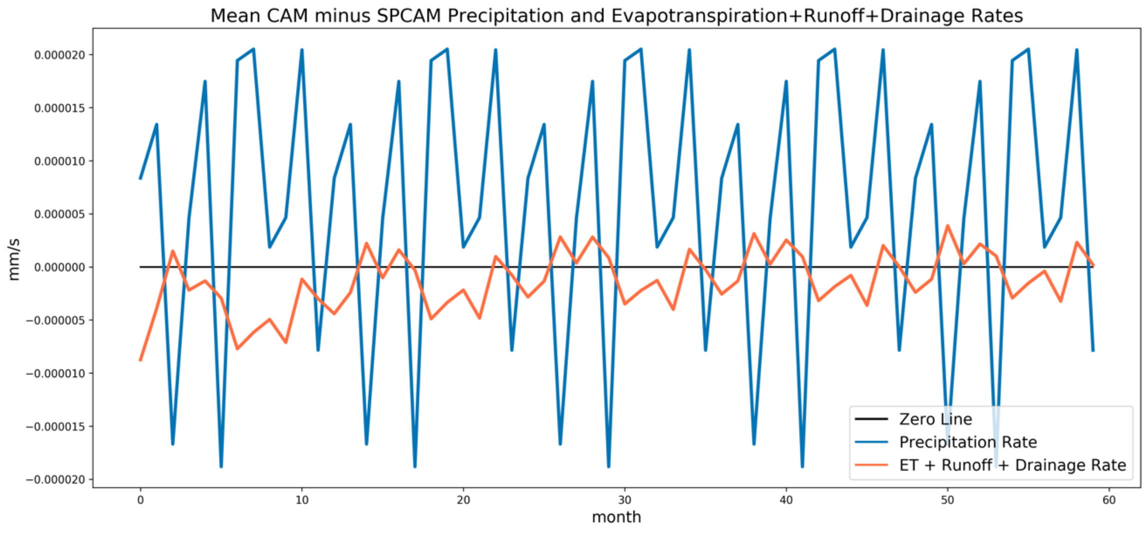
(a)


Volumetric Soil Water -- Soil Depth 2.865m
mm3/mm3
0.44
0.42
0.40
0.38
0.36
0.34
0.32
0.30
CAM
SPCAM
NNCAM
COUPLED
2004
2005
2006
2007
2008
2009
month

(b)


Total Volumetric Soil Water
mm3/mm3
3.8
3.6
3.4
3.2
3.0
2.8
2.6
2.4
CAM
SPCAM
NNCAM
COUPLED
2004
2005
2006
2007
2008
2009
month

(c)


Mean CAM minus SPCAM Precipitation and Evapotranspiration+Runoff+Drainage Rates
mm/s
0.000020
0.000015
0.000010
0.000005
0.000000
−0.000005
−0.000010
−0.000015
−0.000020
Zero Line
Precipitation Rate
ET + Runoff + Drainage Rate
0
10
20
30
40
50
60
month

(d)

Figure 4.8: Volumetric soil water is displayed at (a) the soil surface, (b) 2.8m, and (c) for the soil column as a whole. The meaning of the different colors and lines in the time series is the same as for Figure 4.5. Surface soil water for all aquaplanet models is relatively constant year to year, but the water at increased depths decreases more rapidly for SPCAM than CAM in the first year, and for SPCAM becomes lower than even the seasonal minimum of the coupled run throughout the 5 year simulation. The water content at lower model levels than 2.8m exist in negligible concentrations throughout the simulation. (d) shows that the difference between models for precipitation rate is greater than that of the combined evapotranspiration, runoff, and drainage rates.

Normally the divergent behavior seen in NEE in this experiment might be easily attributed to differences in the GPP levels, as GPP typically drives interannual variability in NEE (Jung et al., 2011). However, the difference in GPP cannot fully explain the large differences in NEE shown each simulation year, which also appear to increase in severity over time. A compounding change in the ecosystem must be taking place. NEE includes land use change, crop harvest, and fire events. Since land use change is disabled during the course of the simulation, and none of the PFTs are crops, fire is a likely suspect. While fires are not commonly observed in the Amazon, they play an important role in ecosystem dynamics; for example, the cycle of infrequent fires during particularly dry seasons in the Amazon enables greater biodiversity (Goldammer, 1992). In the case of this experiment, fires may or may not be even more likely under the aquaplanet atmospheric forcing scenario compared to coupled simulations, which are already more frequent under the simulated fire regime of CLM4 than seen in observations (Thonicke et al., 2001).

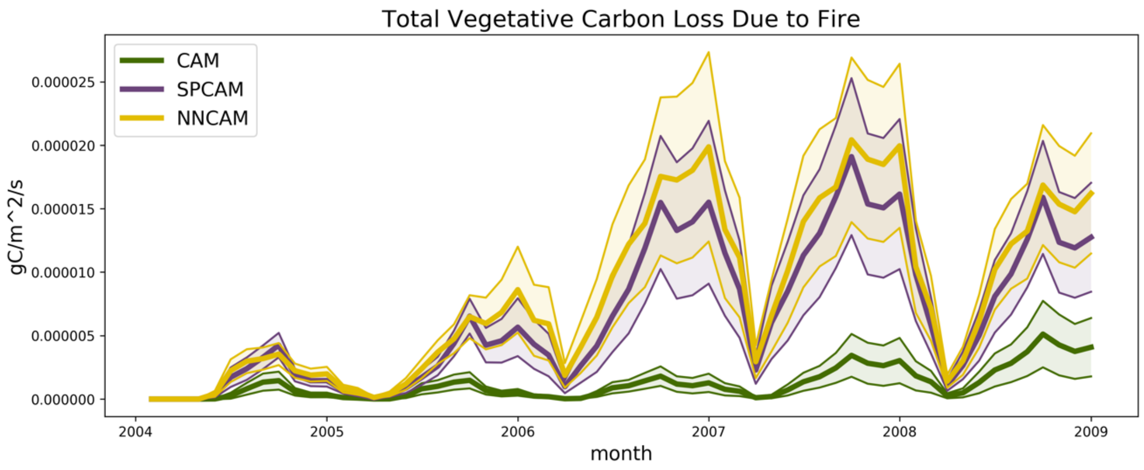
The fire regime in CLM4 uses temperature, organic litter quantity, and litter moisture as factors in determining if a fire event begins, and the subsequent effects of the fire are driven primarily by the length of fire season calculated at each timestep. Knowing this, that there might be a difference in fire probability between CAM and SPCAM makes sense when looking at the atmospheric forcing differences between models: the lower soil moisture and higher temperature may be different enough to impact the number of fires occurring between SPCAM and CAM.

The formula for determining the probability of one fire per day in a land cell is

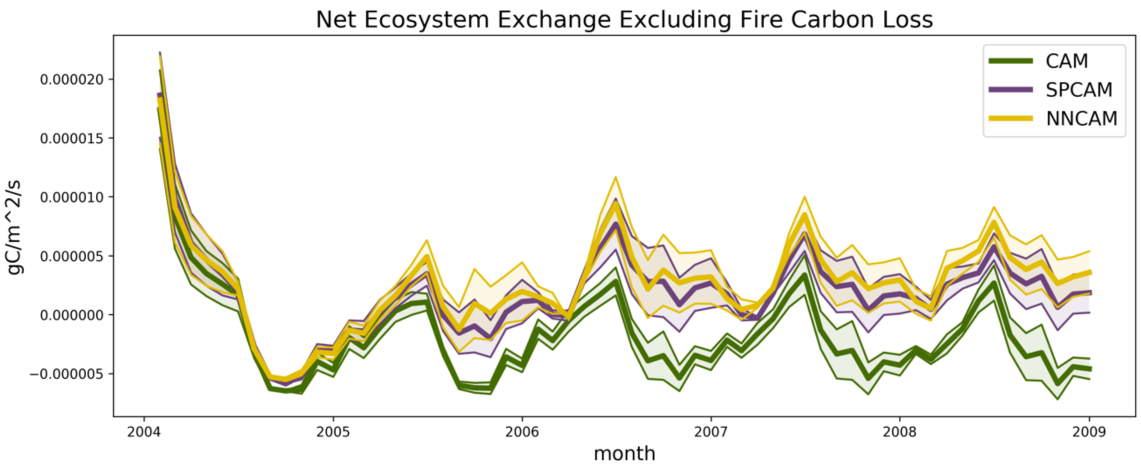
$$p(m) = e^{-\pi\left(\frac{m}{m_e}\right)^2}$$

where *m* is the daily moisture in the upper soil layer and m_e is the "moisture of extinction" (Oleson et al., 2010; Thonicke et al., 2001). The moisture of extinction is the threshold of fuel moisture above which fire cannot spread, and is set at a constant 30% for the 99.9% majority of PFTs on the land cell. As long as the temperature is high enough and enough fuel is present, a fire will occur on any given day of the simulation with probability *p(m)*. The fraction of area of land cell burnt in a fire event is related to the yearly sum of daily probabilities of fire occurrence, and mortality for the tropical trees (99.9% of the land cell) in the simulation is 88% of the fraction burned.

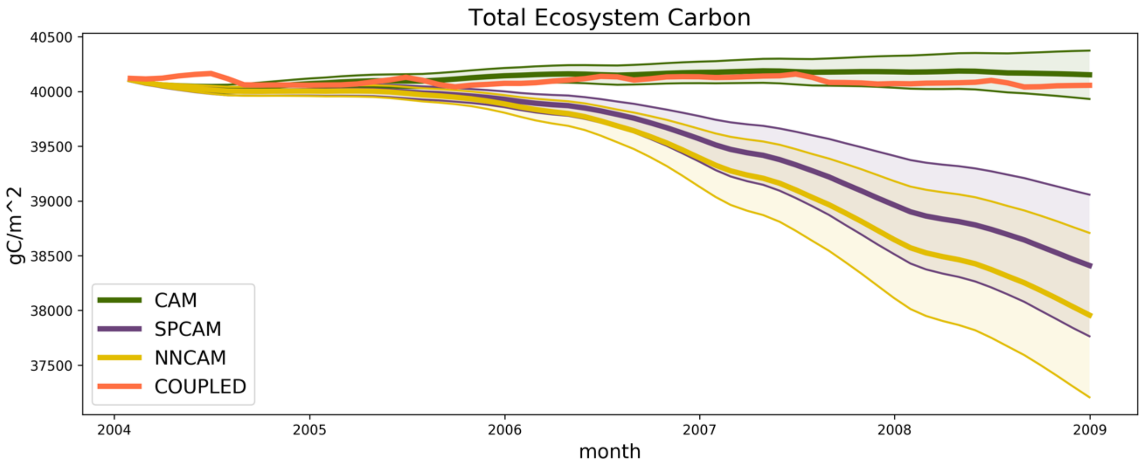
Figure 4.9a reveals that fires are much more significant in the SPCAM simulations than CAM simulations. Because the temperature on any given calendar day of the simulation is the same every year, and for all ensembles a fire is occurring, we can conclude that, at least for a fraction of the cases across the 112 simulation ensemble, lower temperature is not preventing fires in the CAM simulations. Additionally, because the fuel load (litter) is initially the same for every simulation, fuel load limitation is not the cause of difference either. In cases where lower temperature is not preventing fires in the CAM simulations, the primary cause therefore must be mainly due to differences in soil moisture. As the time series progresses, the soil moisture slowly but steadily declines for both CAM and SPCAM, and the litter from fires increase. Both of these factors must contribute to the increasing severity of the fires seen. 4.9b reveals that absent fire NEE is much more similar between the simulations, but it is still larger for SPCAM, which is possibly due to lower overall photosynthetic activity per PFT from reduced soil moisture. Finally, 4.9c confirms that the total ecosystem carbon for SPCAM is decreasing compared to CAM and the coupled simulation. With this evidence, the large difference seen in GPP from Figure 4.4 is almost certainly determined by the fact that there is simply less and less vegetation in the SPCAM simulations than in CAM due to fire.



(a)



(b)



(c)

Figure 4.9: (a) total vegetative carbon loss due to fire, (b) net ecosystem exchange minus the loss of carbon due to fire, and (c) total ecosystem carbon. The thick lines are the means of the simulations and the thin lines are the standard error values surrounding the mean. The coupled simulation data is not shown for (a) and (b), but (c) reveals that CAM's total ecosystem carbon is not significantly different from the coupled simulation after 5 years starting from the same initial conditions.

The 95% confidence that GPP and NEE for SPCAM and NNCAM are lower than CAM, and the rejection of the null hypothesis for statistically insignificant differences between SPCAM/NNCAM and CAM is remarkable given the sensitivity of the simulated biogeochemistry of CLM. That the atmospheric forcing from an online NNCAM aquaplanet simulation would emulate SPCAM enough to produce the same land response and that the three null hypothesis stated before the experimental results would be rejected is not an obvious conclusion at the outset. It was possible that none of the atmospheric forcing variables could have been significantly similar enough between NNCAM and SPCAM to produce similar behavior, which would have been reasonable to predict given the variances seen from modern ESM cloud parameterizations. This is even further impressive considering the arguably most important atmospheric forcing variable in the simulations, precipitation, was directly computed at each time step by the neural network parameterization. As a last note, that mean precipitation levels played such an important role in the outcome of the simulations further cements the significance that SP has in predicting precipitation over traditional cloud parameterizations in regard to its impact on the biosphere.

CHAPTER 5

Summary and Conclusions

Ecosystem dynamics are limited by the complexity of the models that simulate them. For example, despite the extreme biodiversity of the Amazon, biodiversity in CLM4 is limited to 16 plant functional types, and a number of processes are heavily simplified and parameterized, such as the value for the moisture of extinction for all woody plants being identical. However, more modern implementations of CLM are increasing their complexity (Lawrence et al., 2018). By contrast, even though the model complexity exists, the atmospheric component of ESMs are limited by the computational expense required to explicitly calculate physical processes at high enough resolution to accurately capture the effects of those processes, such as deep moisture convection, cloud turbulence, and microphysics. Traditionally the solution to the low resolution ESM problem has been through parameterization, but advances in technology and exploration of machine learning techniques are promising areas of progress in addressing the computational limitations of atmospheric climate simulations (Krasnopolsky & Fox-Rabinovitz, 2005; O'Gorman & Dwyer, 2018; Schneider et al., 2017). Solving the computational limitation challenge of ESMs is critical to predictive capability of land models which are so dependent on the atmospheric conditions they is subject to.

CRMs have been shown to improve prognostic atmospheric simulations, and this paper explored methods of addressing the computational complexity of utilizing CRMs as SP within an ESM. Chapter 2 illustrated the technical challenges of implementing legacy CRM code on cutting-edge hardware, both from a refactoring standpoint and from the perspective of the ever-moving goalpost of advanced computing hardware. Chapters 3 and

4 presented evidence of the efficacy of training neural networks on CRM simulations and using them as parameterizations within CAM simulations, as well as looking at their efficacy in emulating SPCAM when applied to CLM in an idealized experimental setting.

Utilizing neural networks within the context of climate modeling has shown recent successes (Brenowitz & Bretherton, 2018; Gentine et al., 2018; Rasp et al., 2018). However, as seen in this paper the difficulty in enabling a successful NNCAM run in even an idealized aquaplanet simulation highlights the challenge neural networks face in being used in online climate simulations that begin to approach the more realistic settings required for operational climate prediction. It is apparent that thoughtful design is essential for both NN skill and integration with ESMs. Training NNs on SP is an attractive candidate for NN implementation, as SP improves the accuracy of many atmospheric processes and land-atmosphere interactions (Sun & Pritchard, 2016) at the same time that its computational expense is a major roadblock. CBRAIN and NNCAM have shown that NN representation of SP is possible, and the major challenges ahead are likely in design, engineering, and implementation.

A current and future issue that is bound to plague a NNCAM model is generalization. NNCAM did not generalize very well outside of its training bounds (see also Rasp et al. 2018). This is unsurprising as it is common for a NN to overfit its training data and generalize poorly to real-world data. Given the complexity of the Earth System, any NN will at some point be given data it has never seen before. Additionally, ESMs are used for experiments in predicting scenarios under a large variety of conditions that may or may not exist. It would be difficult to train a NN based on every possible scenario, and in the case it were done it is likely that the NN would tend to predict the mean state. Scientists tend to underestimate the likelihood of occurrence and impact of extreme events, and the utilization of a poorly generalized NN would only compound the matter.

That ESMs are intended among other things to study climate *change* makes apparent the issue that training a neural network on current predictions and observations may not lead to good long-term predictions. A neural network trained to be able to recognize the numbers 0-9 can be expected to work as well in the future as it does currently, understanding that 0-9 will look much the same in 50 years as it does now; but a climate model using a neural network trained on current atmospheric simulations and observations cannot be automatically trusted to make a trusted 50-year climate prediction. The idea of training the neural network while the simulation is taking place is a tempting idea, but it would be rather prone to overfitting since it would simply be training on data it produced itself, and further, it would lose out on the advantage of being able to train on highly expensive data simulations up front.

Despite all challenges, climate prediction models that makes use of machine learning techniques such as NNs is likely to occur in the near future. The advantages of machine learning approaches, explored here as the ability to perform higher-resolution simulations without the computational burden, outweigh the difficulty in research and development. NNCAM's SP-parameterization is 20 times faster than SP, and even shows significant improvement in speed over CAM's traditional cloud parameterization. Added to this is the fact that no matter how long or complicated training is, the online implementation will be the same speed. In other words, a cloud process representation scheme far more complex, realistic, and computationally expensive than even SP could be used for training and the online speed of cloud parameterization within NNCAM would be the same. This underscores the real potential of this approach. Training a NN on both observations and high resolution simulations is an attractive idea posed by Schneider et al., 2017, and likely the best next step to be taken in improving NN parameterizations within an ESM alongside advances in NN design, features, preprocessing, and implementation, which would be improved collectively by each research team implementing NN's in their models. How powerful and to what level of widespread approval is required for NN-enhanced simulations of the ecosystem and atmosphere to achieve the level of international acceptance of, for example, the IPCC remains an open question.

Having high resolution cloud-resolving simulations is critical to understanding the complex feedback interactions between the land and atmosphere, particularly with regard to the biosphere where the steady states of vegetative processes depend on the representation of atmospheric radiation, thermodynamics, and the hydrologic and carbon cycles. Coupling the land with an NN-integrated atmospheric model such as NNCAM would better enable higher-resolution ecosystem dynamics experiments and more fully realize the potential of using machine learning in Earth System modeling.

REFERENCES

Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., … Brain, G. (2016). TensorFlow: A System for Large-Scale Machine Learning TensorFlow: A system for large-scale machine learning. In *12th USENIX Symposium on Operating Systems Design and Implementation (OSDI '16)* (pp. 265–284). https://doi.org/10.1038/nn.3331

Ahlström, A., Canadell, J. G., Schurgers, G., Wu, M., Berry, J. A., Guan, K., & Jackson, R. B. (2017). Hydrologic resilience and Amazon productivity. *Nature Communications*, *8*(1), 1–9. https://doi.org/10.1038/s41467-017-00306-z

Allen, C. D., Macalady, A. K., Chenchouni, H., Bachelet, D., McDowell, N., Vennetier, M., … Cobb, N. (2010). A global overview of drought and heat-induced tree mortality reveals emerging climate change risks for forests. *Forest Ecology and Management*, *259*(4), 660–684. https://doi.org/10.1016/j.foreco.2009.09.001

Baker, I. T., Prihodko, L., Denning, A. S., Goulden, M., Miller, S., & Da Rocha, H. R. (2009). Seasonal drought stress in the amazon: Reconciling models and observations. *Journal of Geophysical Research: Biogeosciences*, *114*(1), 1–10. https://doi.org/10.1029/2007JG000644

Baxter, S. M., Day, S. W., Fetrow, J. S., & Reisinger, S. J. (2006). Scientific software development is not an oxymoron. *PLoS Computational Biology*, *2*(9), 0975–0978. https://doi.org/10.1371/journal.pcbi.0020087

Benedict, J. J., & Randall, D. A. (2009). Structure of the Madden–Julian Oscillation in the Superparameterized CAM. *Journal of the Atmospheric Sciences*, *66*(11), 3277–3296. https://doi.org/10.1175/2009JAS3030.1

Bonan, G. B. (2008). Forests and Climate Change : Forcings, Feebacks, and the Climate

Benefits of Forests. *Science*, *320*(June), 1444–1450. https://doi.org/10.1126/science.1155121

Bonan, G. (2015). *Ecological climatology: concepts and applications*. Cambridge University Press.

Brenowitz, N. D., & Bretherton, C. S. (2018). Prognostic Validation of a Neural Network Unified Physics Parameterization. *Geophysical Research Letters*, *45*(12), 6289–6298. https://doi.org/10.1029/2018GL078510

Chambers, J. Q., Higuchi, N., Schimel, J. P., Ferreira, L. V, & Melack, J. M. (2000). Decomposition and carbon cycling of dead trees in tropical forests of the central Amazon. *Oecologia*, *122*(3), 380–388. https://doi.org/10.1007/s004420050044

Chang, K. Y., Paw U, K. T., & Chen, S. H. (2018). The importance of carbon-nitrogen biogeochemistry on water vapor and carbon fluxes as elucidated by a multiple canopy layer higher order closure land surface model. *Agricultural and Forest Meteorology*, *259*(April), 60–74. https://doi.org/10.1016/j.agrformet.2018.04.009

Chapelle, O., Schölkopf, B., & Zien, A. *Semi-Supervised Learning*.

Chen, J., Chen, B., Black, T. A., Innes, J. L., Wang, G., Kiely, G., … Wohlfahrt, G. (2013). Comparison of terrestrial evapotranspiration estimates using the mass transfer and Penman-Monteith equations in land surface models. *Journal of Geophysical Research: Biogeosciences*, *118*(4), 1715–1731. https://doi.org/10.1002/2013JG002446

Chollet, F. (2015). Keras.

Cox, P. M., Betts, R. A., Jones, C. D., Spall, S. A., & Totterdell, I. J. (2000). Acceleration of global warming due to carbon-cycle feedbacks in a coupled climate model (vol 408, pg 184, 2000). *Nature*, *408*(6813), 750. https://doi.org/10.1038/35041539 da Costa, C. L., Galbraith, D., Almeida, S., Tanaka Portela, B. T., da Costa, M., de Athaydes Silva Junior, J., … Meir, P. (2010). Effect of seven years of experimental drought on the aboveground biomass storage of an eastern Amazonian rainforest. *New Phytologist*, *187*, 579–591. https://doi.org/10.1111/j.1469-8137.2010.03309.x

Fausett, L. V. (1994). *Fundamentals of neural networks: architectures, algorithms, and applications* (Vol. 3). Englewood Cliffs: Prentice-Hall.

Flato, G., Marotzke, J., Abiodun, B., Braconnot, P., Chou, S. C., Collins, W., … Rummukainen, M. (2013). Evaluation of Climate Models. *Climate Change 2013: The Physical Science Basis. Contribution of Working Group I to the Fifth Assessment Report of the Intergovernmental Panel on Climate Change*, 741–866. https://doi.org/10.1017/CBO9781107415324

Friedlingstein, P., Meinshausen, M., Arora, V. K., Jones, C. D., Anav, A., Liddicoat, S. K., & Knutti, R. (2014). Uncertainties in CMIP5 climate projections due to carbon cycle feedbacks. *Journal of Climate*, *27*(2), 511–526. https://doi.org/10.1175/JCLI-D-12-00579.1

Gamma, E., Helm, R., Johnson, R., & Vlissides, J. (2002). Design Patterns – Elements of Reusable Object-Oriented Software. *A New Perspective on Object-Oriented Design*, 334. https://doi.org/10.1093/carcin/bgs084

Gentine, P., Pritchard, M., Rasp, S., Reinaudi, G., & Yacalis, G. (2018). Could machine learning break the convection parameterization deadlock? *Geophysical Research Letters*, 1–10. https://doi.org/10.1029/2018GL078202

Glorot, X., & Bengio, Y. (2010). Understanding the difficulty of training deep feedforward neural networks. *PMLR*, *9*, 249–256. https://doi.org/10.1.1.207.2059

Goodfellow, I., Pouget-Abadie, J., … M. M.-A. in neural, & 2014, U. (2014). Generative adversarial nets. *Advances in Neural Information Processing Systems*, 2672–2680. https://doi.org/10.1017/CBO9781139058452

Grossberg, S. (2013). Recurrent neural networks. *Scholarpedia*, 8(2), 1888.

Hochreiter, S., & Schmidhuber, J. (1997). Long Short-Term Memory. *Neural Computation*, *9*(8), 1735–1780. https://doi.org/10.1162/neco.1997.9.8.1735

Huete, A. R., Didan, K., Shimabukuro, Y. E., Ratana, P., Saleska, S. R., Hutyra, L. R., … Myneni, R. (2006). Amazon rainforests green-up with sunlight in dry season. *Geophysical Research Letters*, *33*(6), 2–5. https://doi.org/10.1029/2005GL025583

Hunt, A. (1999). *The pragmatic programmer*. Pearson Education India

Jeffers, J., Reinders, J., & Sodani, A. (2016a). Knights Landing architecture. In *Intel Xeon Phi Processor High Performance Programming* (2nd ed., pp. 63–84). Elsevier Inc. https://doi.org/10.1016/B978-0-12-809194-4.00004-1

Jeffers, J., Reinders, J., & Sodani, A. (2016b). Programming MCDRAM and Cluster modes. In *Intel Xeon Phi Processor High Performance Programming* (2nd ed., pp. 25–61). Elsevier Inc. https://doi.org/10.1016/B978-0-12-809194-4.00003-X

Jeffers, J., Reinders, J., & Sodani, A. (2016c). Weather research and forecasting (WRF). In *Intel Xeon Phi Processor High Performance Programming* (pp. 499–510). https://doi.org/10.1016/B978-0-12-809194-4.00022-3

Jung, M., Reichstein, M., Margolis, H. A., Cescatti, A., Richardson, A. D., Arain, M. A., … Williams, C. (2011). Global patterns of land-atmosphere fluxes of carbon dioxide, latent heat, and sensible heat derived from eddy covariance, satellite, and meteorological observations. *Journal of Geophysical Research: Biogeosciences*, *116*(3), 1–16. https://doi.org/10.1029/2010JG001566

Jung, M., et al. (2010), Recent decline in the global land evapotranspiration trend due to limited moisture supply, *Nature*, 467(7318), 951–954.

Karpathy, A., Toderici, G., Shetty, S., Leung, T., Sukthankar, R., & Li, F. F. (2014). Large-scale video classification with convolutional neural networks. *Proc. IEEE CVPR*, 1725–1732.

Katul, G. G., R. Oren, S. Manzoni, C. Higgins, and M. B. Parlange (2012), Evapotranspiration: A process driving mass transport and energy exchange in the soil-plant-atmosphere-climate system, *Reviews of Geophysics*, 50, RG3002, doi:10.1029/2011RG000366

https://doi.org/10.1109/CVPR.2014.223

Keogh, E., & Ratanamahatana, C. A. (2005). Exact indexing of dynamic time warping. *Knowledge and Information Systems*, *7*(3), 358–386. https://doi.org/10.1007/s10115-004-0154-9

Khairoutdinov, M. F., & Randall, D. A. (2003). Cloud Resolving Modeling of the ARM Summer 1997 IOP: Model Formulation, Results, Uncertainties, and Sensitivities. *Journal of the Atmospheric Sciences*, *60*(4), 607–625. https://doi.org/10.1175/1520-0469(2003)060<0607:CRMOTA>2.0.CO;2

Khairoutdinov, M. (2014). System for Atmospheric Modeling Version 6.10.6 User's Guide.

Kianimajd, A., Ruano, M. G., Carvalho, P., Henriques, J., Rocha, T., Paredes, S., & Ruano, A. E. (2017). Comparison of different methods of measuring similarity in physiologic time series. *IFAC-PapersOnLine*, *50*(1), 11005–11010. https://doi.org/10.1016/j.ifacol.2017.08.2479

Kiehl, J. T., Trenberth, K. E., Kiehl, J. T., & Trenberth, K. E. (1997). Earth's Annual Global Mean Energy Budget. *Bulletin of the American Meteorological Society*, *78*(2), 197–208.

https://doi.org/10.1175/1520-0477(1997)078<0197:EAGMEB>2.0.CO;2

Kiyoshi Kawaguchi. *A multithreaded software model for backpropagation neural network applications*. 2000.

Kooperman, G. J., Pritchard, M. S., Burt, M. A., Branson, M. D., & Randall, D. A. (2016a). Impacts of cloud superparameterization on projected daily rainfall intensity climate changes in multiple versions of the Community Earth System Model. *Journal of Advances in Modeling Earth Systems*, *8*, 1727–1750. https://doi.org/10.1002/2016MS000660

Kooperman, G. J., Pritchard, M. S., Burt, M. A., Branson, M. D., & Randall, D. A. (2016b). Robust effects of cloud superparameterization on simulated daily rainfall intensity statistics across multiple versions of the Community Earth System Model. *Journal of Advances in Modeling Earth Systems*, *8*(1), 140–165. https://doi.org/10.1002/2015MS000574

Krasnopolsky, V. M., & Fox-Rabinovitz, M. S. (2005). Complex hybrid models combining deterministic and machine learning components as a new synergetic paradigm in numerical climate modeling and weather prediction. *Proceedings of the International Joint Conference on Neural Networks*, *3*, 1615–1620. https://doi.org/10.1109/IJCNN.2005.1556120

Krishnamurthy, V., & Stan, C. (2015). Simulation of the South American climate by a coupled model with super-parameterized convection. *Climate Dynamics*, *44*(9–10), 2369–2382. https://doi.org/10.1007/s00382-015-2476-6

Lawrence, D. M., Oleson, K. W., Flanner, M. G., Thornton, P. E., Swenson, S. C., Lawrence, P. J., … Slater, A. G. (2011). Parameterization improvements and functional and structural advances in Version 4 of the Community Land Model. *Journal of Advances in Modeling Earth Systems*, *3*(1), n/a-n/a. https://doi.org/10.1029/2011MS00045

Lawrence, D. M., Thornton, P. E., Oleson, K. W., & Bonan, G. B. (2007). The Partitioning of Evapotranspiration into Transpiration, Soil Evaporation, and Canopy Evaporation in a GCM: Impacts on Land–Atmosphere Interaction. *Journal of Hydrometeorology*, *8*(4), 862–880. https://doi.org/10.1175/JHM596.1

Lawrence et al., (2018). Technical Description of version 5.0 of the Community Land Model (CLM), (February).

Lhermitte, S., Verbesselt, J., Verstraeten, W. W., & Coppin, P. (2011). A comparison of time series similarity measures for classification and change detection of ecosystem dynamics. *Remote Sensing of Environment*, *115*(12), 3129–3152. https://doi.org/10.1016/j.rse.2011.06.020

Li, W., Fu, R., & Dickinson, R. E. (2006). Rainfall and its seasonality over the Amazon in the 21st century as assessed by the coupled models for the IPCC AR4. *Journal of Geophysical Research Atmospheres*, *111*(2), 1–14. https://doi.org/10.1029/2005JD006355

Lipton, Z. C. (2015). A Critical Review of Recurrent Neural Networks for Sequence Learning. *CoRR*, *abs/1506.0*, 1–38. https://doi.org/10.1145/2647868.2654889

Maas, A. L., Hannun, A. Y., & Ng, A. Y. (2013). Rectifier Nonlinearities Improve Neural Network Acoustic Models. *Proceedings of the 30 Th International Conference on Machine Learning*, *28*, 6. Retrieved from https://web.stanford.edu/~awni/papers/relu_hybrid_icml2013_final.pdf

Maddison, C. J., Huang, A., Sutskever, I., & Silver, D. (2014). Move Evaluation in Go Using

Deep Convolutional Neural Networks. *Http://Arxiv.Org/Abs/1412.6564*, 1–8. Retrieved from http://arxiv.org/abs/1412.6564

Malhi, Y., Roberts, J. T., Betts, R. a, Killeen, T. J., Li, W., & Nobre, C. a. (2008). Climate Change, Deforestation, and the Fate of the Amazon. *Science*, *319*(iv), 169–172. https://doi.org/10.1126/science.1146961

McMahon, S. M., Harrison, S. P., Armbruster, W. S., Bartlein, P. J., Beale, C. M., Edwards, M. E., … Prentice, I. C. (2011). Improving assessment and modelling of climate change impacts on global terrestrial biodiversity. *Trends in Ecology and Evolution*, *26*(5), 249–259. https://doi.org/10.1016/j.tree.2011.02.012

Medeiros, B., Williamson, D. L., & Olson, J. G. (2016). Reference aquaplanet climate in the Community Atmosphere Model, Version 5. *Journal of Advances in Modeling Earth Systems*, *8*(1), 406–424. https://doi.org/10.1002/2015MS000593

Miralles, D. G., Gentine, P., Seneviratne, S. I., & Teuling, A. J. (2018). Land-atmospheric feedbacks during droughts and heatwaves: state of the science and current challenges. *Annals of the New York Academy of Sciences*, 1–17. https://doi.org/10.1111/nyas.13912

Nair, V., & Hinton, G. E. (2010). Rectified Linear Units Improve Restricted Boltzmann Machines. *Proceedings of the 27th International Conference on Machine Learning*, (3), 807–814. https://doi.org/10.1.1.165.6419

Nepstad, D. C., De Carvalho, C. R., Davidson, E. A., Jipp, P. H., Lefebvre, P. A., Negreiros, G. H., … Vieira, S. (1994). The role of deep roots in the hydrological and carbon cycles of Amazonian forests and pastures. *Nature*. https://doi.org/10.1038/372666a0

Nielsen, Michael A. (2015). *Neural Networks and Deep Learning*. Determination Press.

O'Gorman, P. A., & Dwyer, J. G. (2018). Using machine learning to parameterize moist convection: potential for modeling of climate, climate change and extreme events, 1–20. Retrieved from http://arxiv.org/abs/1806.11037

Oki, T., and S. Kanae. (2006), Global hydrological cycles and world water re- sources, *Science*, 313(5790), 1068–1072.

Oleson, K. W., Lawrence, D. M., Gordon, B., Flanner, M. G., Kluzek, E., Peter, J., … Zeng, X. (2010). Technical Description of version 4 . 0 of the Community Land Model (CLM), (April).

Pan, Y., Birdsey, R. A., Fang, J., Houghton, R., Kauppi, P. E., Kurz, W. A., … Hayes, D. (2011). A large and persistent carbon sink in the world's forests. *Science*, *333*(6045), 988–993. https://doi.org/10.1126/science.1201609

Parishani, H., Pritchard, M. S., Bretherton, C. S., Wyant, M. C., & Khairoutdinov, M. (2017). Toward low-cloud-permitting cloud superparameterization with explicit boundary layer turbulence. *Journal of Advances in Modeling Earth Systems*, *9*(3), 1542–1571. https://doi.org/10.1002/2017MS000968

Powell, T. L., Galbraith, D. R., Christoffersen, B. O., Harper, A., Imbuzeiro, H. M. A., Rowland, L., … Moorcroft, P. R. (2013). Confronting model predictions of carbon fluxes with measurements of Amazon forests subjected to experimental drought. *New Phytologist*, *200*(2), 350–365. https://doi.org/10.1111/nph.12390

Qin, H., Pritchard, M. S., Kooperman, G. J., & Parishani, H. (2018). Global Effects of Superparameterization on Hydrothermal Land-Atmosphere Coupling on Multiple Timescales. *Journal of Advances in Modeling Earth Systems*, *10*(2), 530–549. https://doi.org/10.1002/2017MS001185

Randall, B. Y. D., Khairoutdinov, M., Arakawa, A., & Grabowski, W. (2003). Breaking the Cloud Deadlock. *Bulletin of the American Meteorological Society*, (March). https://doi.org/10.1175/BAMS-84-11-1547

Randall, D., DeMott, C., Stan, C., Khairoutdinov, M., Benedict, J., McCrary, R., … Branson, M. (2016). Simulations of the Tropical General Circulation with a Multiscale Global Model. *Meteorological Monographs - Multiscale Convection-Coupled Systems in the Tropics: A Tribute to Dr. Michio Yanai*, 1–15. https://doi.org/10.1175/AMSMONOGRAPHS-D-15-0016.1

Rasp, S., Pritchard, M. S., & Gentine, P. (2018). Deep learning to represent sub-grid processes in climate models, 1–14. Retrieved from http://arxiv.org/abs/1806.04731

Saleska, S. R., Didan, K., Huete, A. R., & Da Rocha, H. R. (2007). Amazon forests green-up during 2005 drought. *Science*, *318*(5850), 612. https://doi.org/10.1126/science.1146663

Schalkoff, R. J. (1997). *Artificial neural networks* (Vol. 1). New York: McGraw-Hill.

Schmidhuber, J. (2015). Deep Learning in neural networks: An overview. *Neural Networks*. Elsevier Ltd. https://doi.org/10.1016/j.neunet.2014.09.003

Schneider, T., Lan, S., Stuart, A., & Teixeira, J. (2017). Earth System Modeling 2.0: A Blueprint for Models That Learn From Observations and Targeted High-Resolution Simulations. *Geophysical Research Letters*, *44*(24), 12,396-12,417. https://doi.org/10.1002/2017GL076101

Sellers, P., Shukla, J., & Nobre, C. (1990). Amazon Deforestation and Climate Change. *Science*, *247*(March), 1322–1325. https://doi.org/10.1126/science.247.4948.1322

Sharma, M., Pachori, R. B., & Rajendra Acharya, U. (2017). ADAM: A METHOD FOR

STOCHASTIC OPTIMIZATION. *Pattern Recognition Letters*, *94*, 172–179. https://doi.org/10.1016/j.patrec.2017.03.023

Solomon, S. (1999). Stratospheric ozone depletion: A review of concepts and history. *Reviews of Geophysics*, *37*(3), 275–316. https://doi.org/10.1029/1999RG900008

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *Journal of Machine Learning Research*, *15*, 1929–1958. https://doi.org/10.1214/12-AOS1000

Stephens, G. L., Li, J., Wild, M., Clayson, C. A., Loeb, N., Kato, S., … Andrews, T. (2012). An update on Earth's energy balance in light of the latest global observations. *Nature Geoscience*, *5*(10), 691–696. https://doi.org/10.1038/ngeo1580

Sun, J., & Pritchard, M. S. (2016). Effects of explicit convection on global land-atmosphere coupling in the superparameterized CAM. *Journal of Advances in Modeling Earth Systems*, *8*(12 AUG 2016), 1248–1269. https://doi.org/10.1002/2016MS000660

Texas Advanced Computing Center. Stampede2 User Guide. (2018, June 13). Retrieved on 2018, August 12 from https://portal.tacc.utexas.edu/user-guides/stampede2

Thayer-Calder, K., & Randall, D. A. (2009). The Role of Convective Moistening in the Madden–Julian Oscillation. *Journal of the Atmospheric Sciences*, *66*(11), 3297–3312. https://doi.org/10.1175/2009JAS3081.1

Thompson, I., Mackey, B., McNulty, S., & Mosseler, A. (2009). *Forest resilience, biodiversity, and climate change* (Vol. 43). Secretariat of the Convention on Biological Diversity Cover.

Thonicke, K., Venevsky, S., Sitch, S., & Cramer, W. (2001). The role of fire disturbance for global vegetation dynamics: coupling fire into a Dynamic Global Vegetation Model.

Global Ecology & Biogeography, *10*(6), 661–677. https://doi.org/10.1046/j.1466-822X.2001.00175.x

Trader, T, (2018, July 25). Requiem for a Phi: Knights Landing Discontinued. Retrieved on 2018, August 12 from https://www.hpcwire.com/2018/07/25/end-of-the-road-for-knights-landing-phi/

Trenberth, K. ~E., Fasullo, J. ~T., & Kiehl, J. (2009). Earth's Global Energy Budget. *Bull. Am. Meteorol. Soc.*, *90*, 311–323. https://doi.org/10.1175/2008BAMS2634.1

Vogt, K. A., Vogt, D. J., Palmiotto, P. A., Boon, P., O'Hara, J., & Asbjornsen, H. (1995). Review of root dynamics in forest ecosystems grouped by climate, climatic forest type and species. *Plant and Soil: An International Journal on Plant-Soil Relationships*, *187*(2), 159–219. https://doi.org/10.1007/BF00017088

Wang, H., & Raj, B. (2017). On the Origin of Deep Learning, 1–72. https://doi.org/10.1139/f56-020

Wofsy, S. C., Zhang, X., Qin, D., Manning, M., Chen, Z., Marquis, M., & Averyt, K. B. (2007). Couplings Between Changes in the Climate System and Biogeochemistry. *Climate Change 2007: The Physical Science Basis*, *21*(7), 499–587. https://doi.org/Cited By (since 1996) 525\rExport Date 12 August 2012

Xu, B., Wang, N., Chen, T., & Li, M. (2015). Empirical Evaluation of Rectified Activations in Convolutional Network. Retrieved from http://arxiv.org/abs/1505.00853

Zeiler, M. D., Ranzato, M., Monga, R., Mao, M., Yang, K., Le, Q. V, … Hinton, G. E. (2013). On rectified linear units for speech processing. In *ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing - Proceedings* (pp. 3517–3521). https://doi.org/10.1109/ICASSP.2013.6638312

Zhang, K., Rong Fu, Shaikh, M. J., Ghan, S., Wang, M., Leung, L. R., … Marengo, J. (2017). Influence of Superparameterization and a Higher-Order Turbulence Closure on Rainfall Bias Over Amazonia in Community Atmosphere Model Version 5. *Journal of Geophysical Research : Atmospheres*, *122*(18), 9879–9902. https://doi.org/10.1002/2017JD026576

APPENDIX A

Euclidean Distance and Dynamic Time Warping

Comparisons between the Euclidean distances of GPP and NEE for CAM, SPCAM, and NNCAM before and after normalization among the means of the time series are here performed. One other time series similarity measure, dynamic time warping (DTW), is also used. DTW measures the similarity of time series differently in that it pares out similar patterns wihtin time series even if those patterns are out of phase (unaligned in time), in different value ranges, and are of unequal length (Keogh & Ratanamahatana, 2005; Kianimajd et al., 2017).

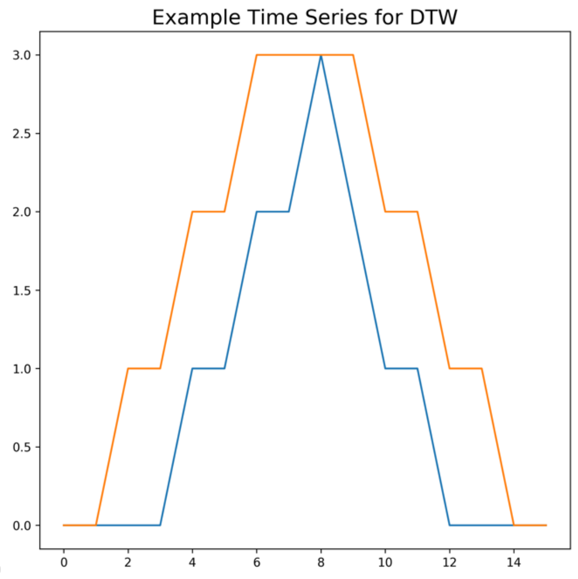
The formula for Eucliden Distance (RMSE) for time series *Q* and *C* from timestep 1 to *n* is:

$$\mathrm{D(Q,C)} = \sqrt{\sum_{\mathrm{i=1}}^{\mathrm{n}} (\mathrm{q_i} - \mathrm{c_i})^2}$$

By contrast, DTW begins by constructing a matrix of distances between the values of *Q* and *C* at every timestep, in this experiment $(q_i - c_j)^2$. A warping path is a mapping along the matrix from the lower left point (1, 1) to the upper right point (*m, n*) of the matrix (see Figure A1). DTW uses a recurrent algorithm to search within the matrix for the warping path *P*, where p_k is the kth element of *P* and has an associated cost (distance between q_i and c_j), that minimizes the cost. *P* is constrained to start at (1, 1) and monotonically increase towards (*m, n*). The function that minimizes the cost is defined by:

$$\mathrm{DTW(Q,C)} = \min\left\{\sqrt{\sum_{\mathrm{k=1}}^{\mathrm{K}} \mathrm{p_k}}\right.$$

Using RMSE on non-normalized time series, RMSE on normalized time series, and DTW all in conjunction are adequate tests to evaluate the relative similarity of the GPP and NEE time series among CAM, SPCAM, and NNCAM, as they calculate absolute similarity; absolute similarity agnostic to offset, amplitude, and linear trend; and similarity agnostic to range and phase. A visual representation of RMSE and DTW can be seen in Figure A2.

(a)


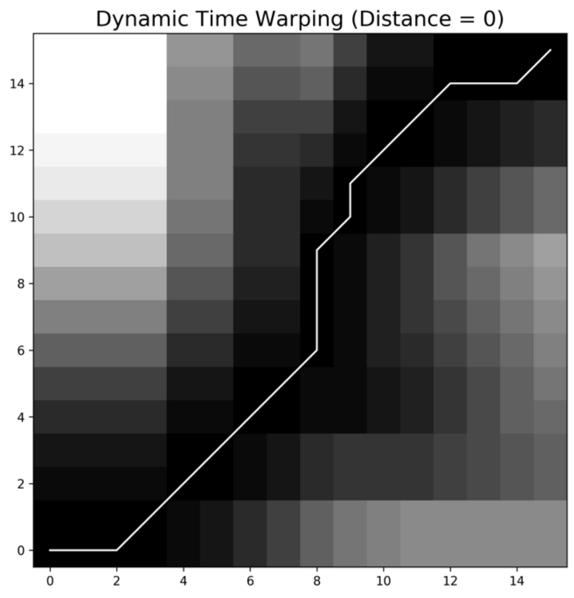
(b)


Figure A1: For the two time series Q (orange) and C (blue) in (a), dynamic time warping (DTW) involves creating the matrix in (b) and finding the path from the beginning of the time series (1, 1) to the end (*m, n*) that has the lowest cost, where each point in the matrix has a cost computed by $\left(q_i - c_j\right)^2$. The whitest tiles of the matrix have the highest cost, and the blackest tiles have the lowest cost. In this case, a path was found that has a total cost of 0, so the DTW distance value produced is equal to 0.

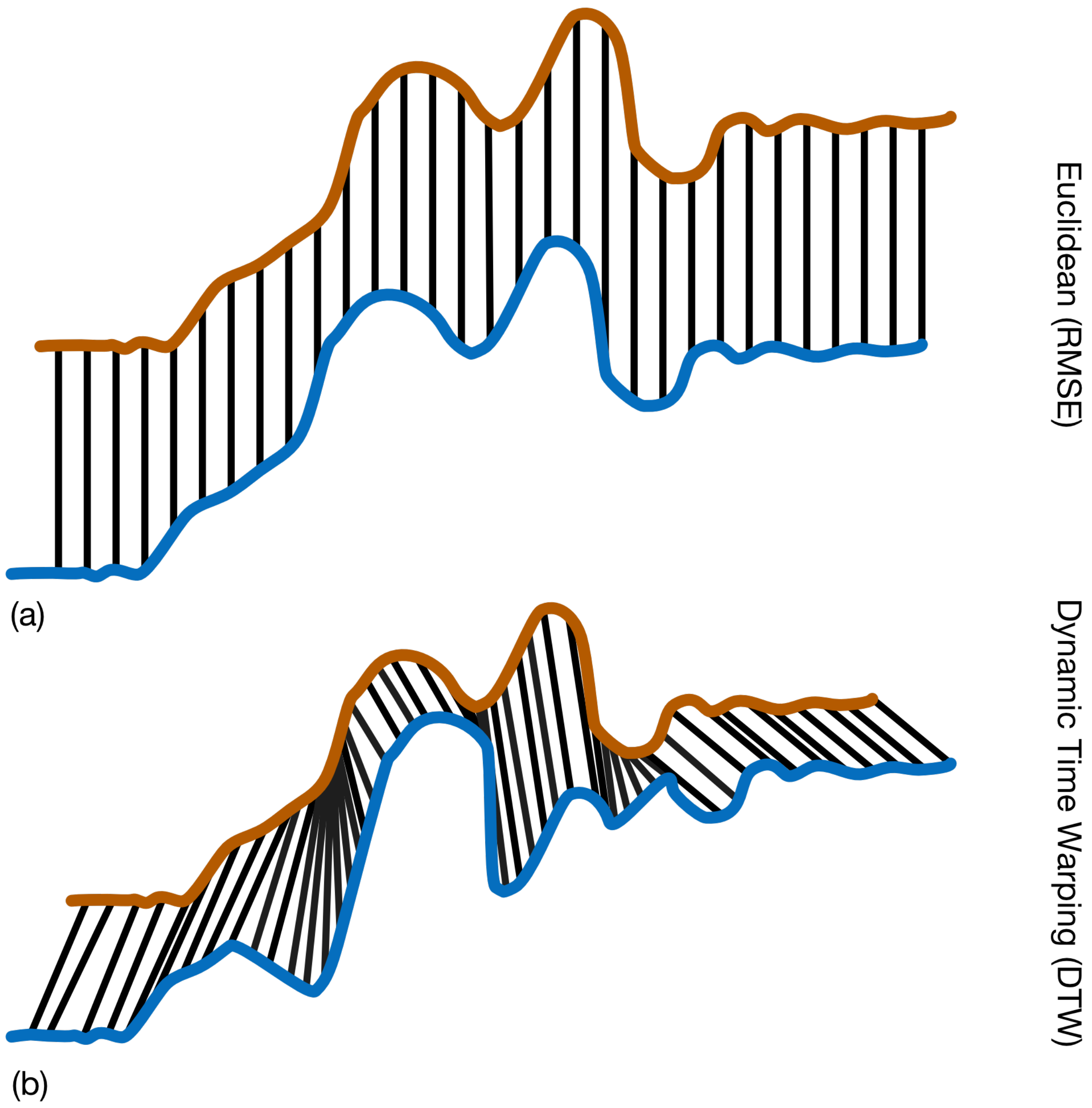
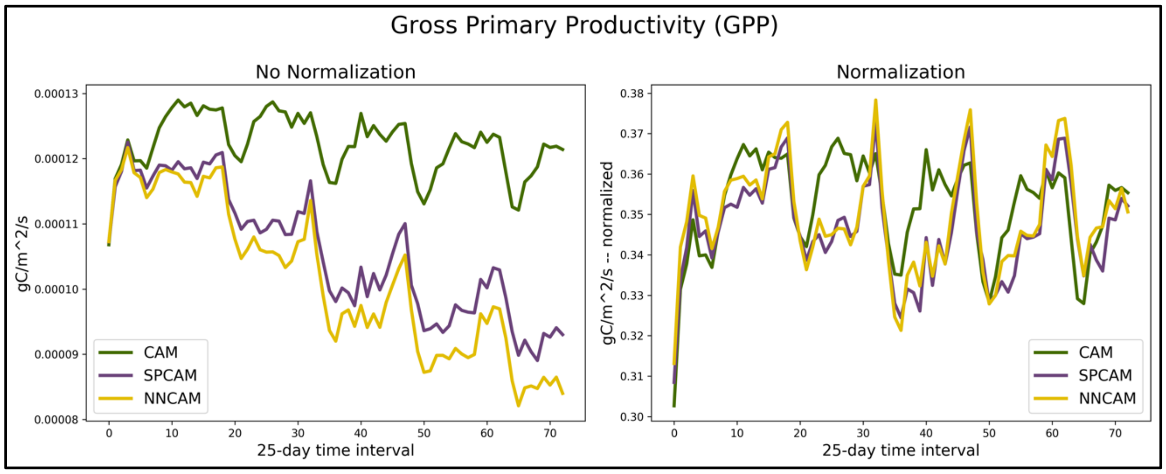


Figure A2: Visual representation of the difference in approaches between (a) Euclidean distance (RMSE) and (b) Dynamic Time Warping (DTW). RMSE focuses in on similarity between values at equivalent timesteps, whereas DTW measures similarity of time series patterns irrespective of distance and phase.

Normalizing the three time series takes a look to see if the local response at a particular time point among time series is similar even if the mean, amplitude, and linear trend are different. Figure A3 visually illustrates the effects of normalization. From the original values, offset is performed by subtracting the mean; dividing by the standard deviation of the original values then normalizes the amplitudes of the time series; finally, performing a linear regression on the resulting time series data (the data values after accounting for offset and amplitude) and subtracting it the gives the fully normalized version of the time series.

(a) 

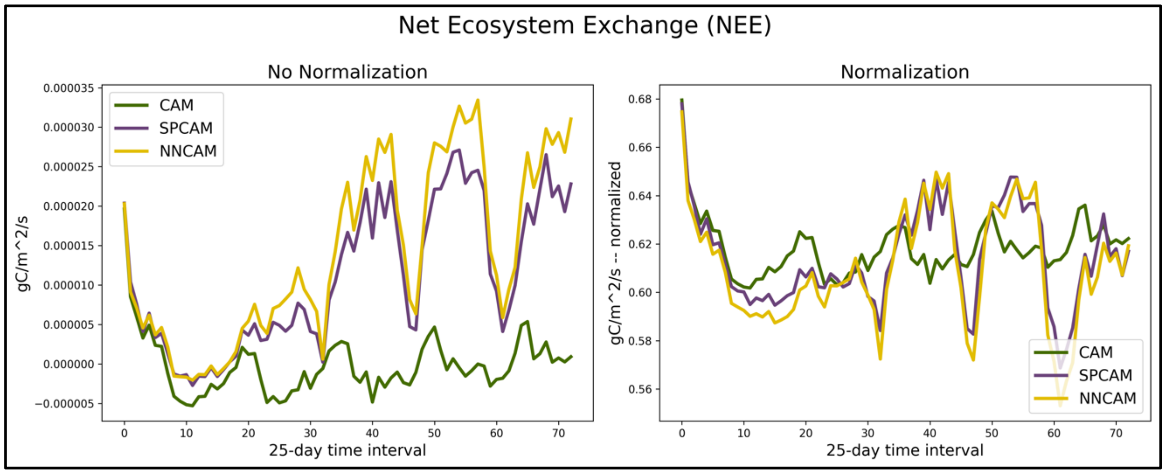
(b)


Figure A3: Visualization of (a) GPP and (b) NEE 25-day means before and after normalization. Normalization includes reducing the offset, standardizing the amplitude, and removing the linear trend. The calculations to create a normalized time series $\hat{x}$ from time series x is $\hat{x} = \frac{mean(x)}{std(x)} - LinearRegressionFit\left(\frac{mean(x)}{std(x)}\right)$. Even after the transformations, NNCAM and SPCAM are more similar to each other than to CAM at a local level.