

Image Detection Using a Pre-Trained Model



A faint, horizontal landscape background featuring snow-capped mountains under a cloudy sky.

Katie Dagon

NSF National Center for Atmospheric Research

ML Bootcamp
November 6, 2025

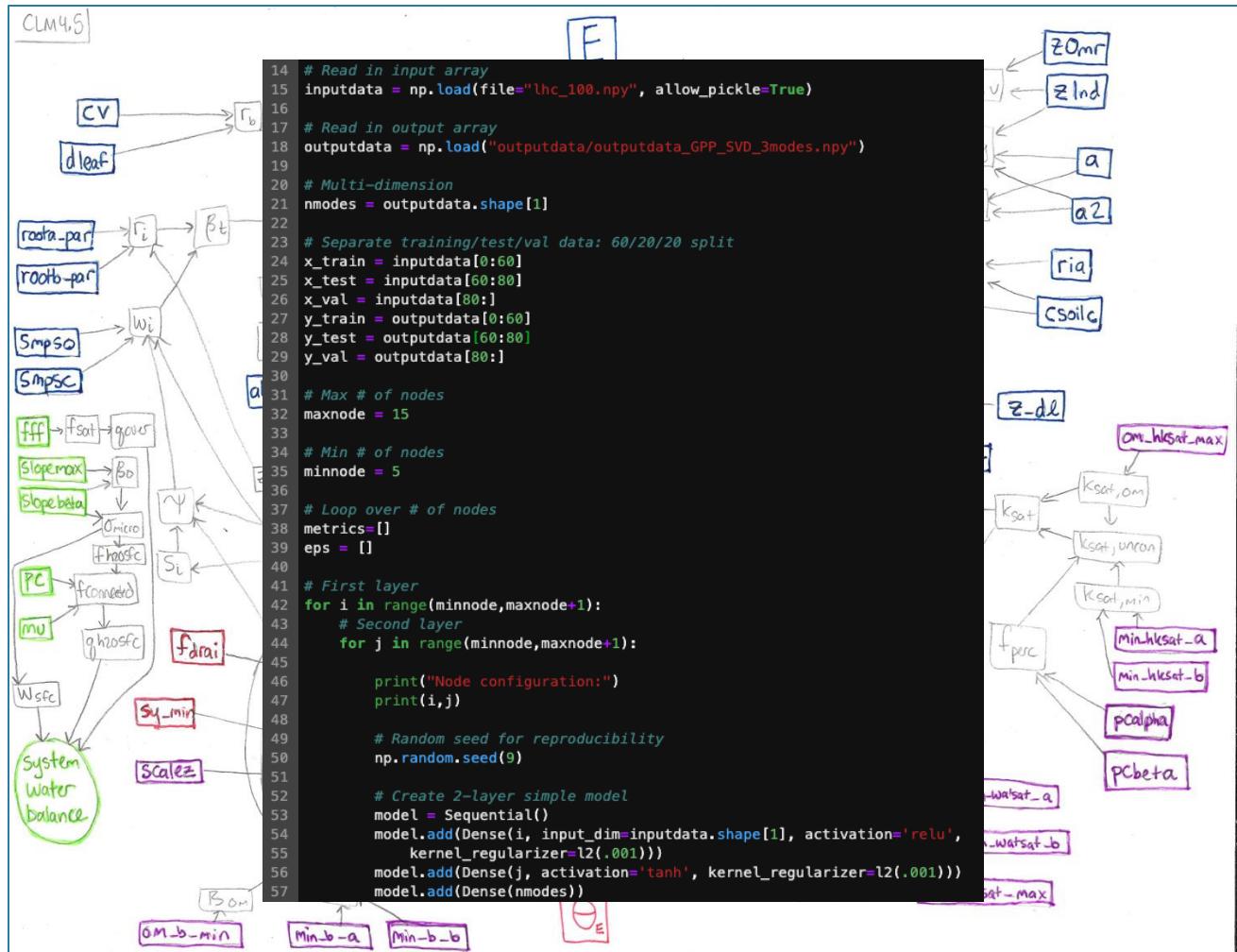
How did I get here?

Exploring land model parameter uncertainty

- How do we fully explore the parameter space of the model without running 1000s of simulations?
- Train ML emulators to map the parameter response space!
- Creating an emulator meant learning some sort of neural network tool...which meant learning Python and Keras



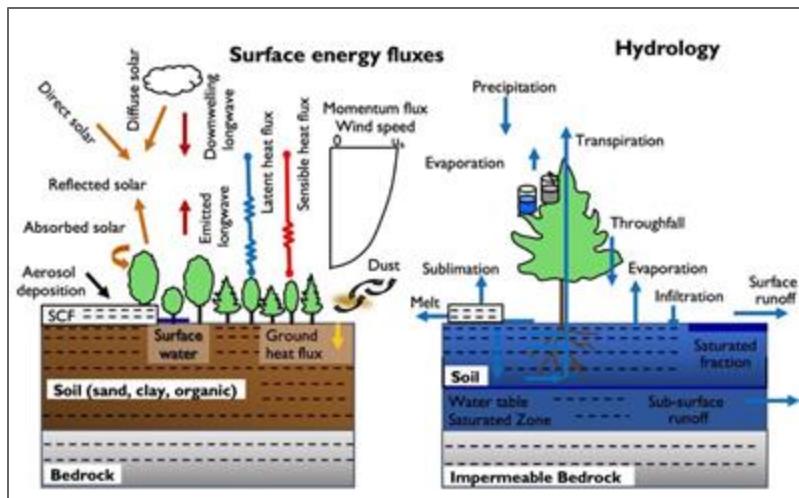
TensorFlow



https://github.com/katiedagon/CLM5_ParameterUncertainty

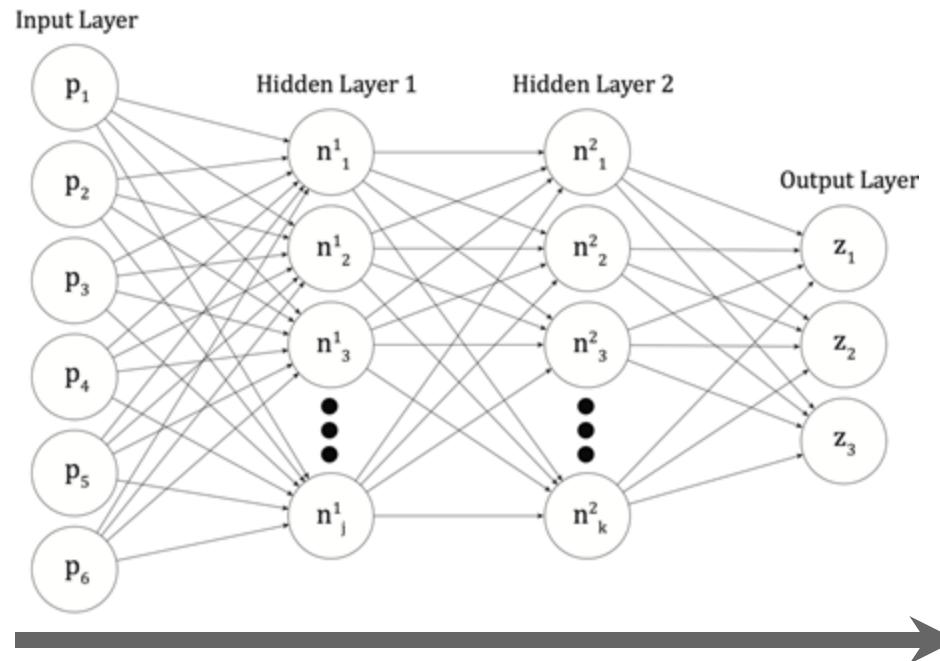
Machine Learning for Land Model Emulation

Input: land model parameter values

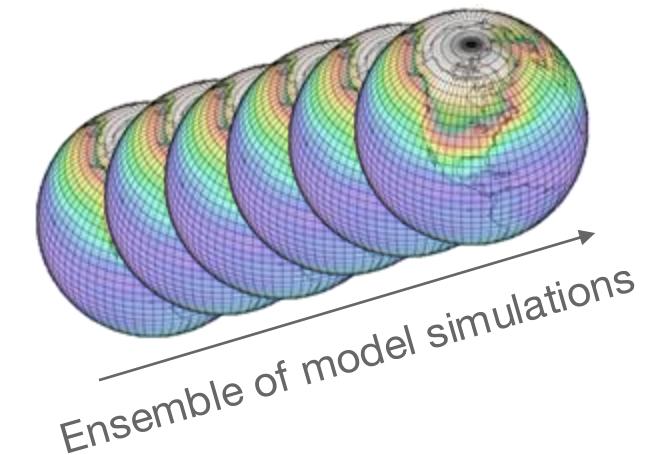


Community Land Model (CLM)

*Neural network emulator:
ANN with 2 layers*

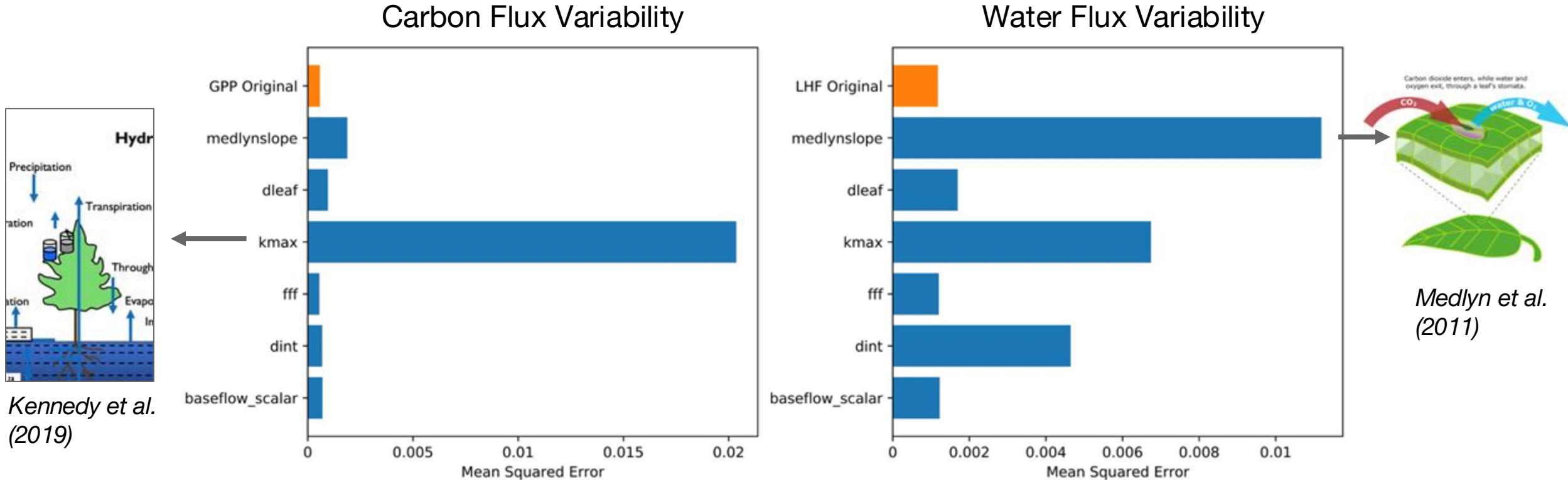


Output: land model
quantities from a
perturbed parameter
ensemble (PPE)



Dagon et al. (2020)

Interpretability Sheds Light on Physics



Variable Importance

- Randomly shuffle values of one parameter (preserving others) and **test performance of emulator**.
- Skill metric is mean squared error between predictions and actual values.
- Larger bar means the parameter is **more important to the predictive skill** of the emulator.

Dagon et al. (2020)

Not Reinventing the Wheel



Model Interpretation

One of the common critiques of machine learning models is that many appear to be a "black box", in which predictions are made without associated reasoning why they were made. However, there are ways to interrogate machine learning models to extract information about how the model weights each input and how the model encodes information internally.

Permutation variable importance is a model-agnostic way of estimating how much each input affects the prediction values on average. In permutation variable importance, a metric is calculated on the model predictions of the validation data. Then each validation set input is permuted, or shuffled, among the examples, and the metric is recalculated. A large change in the metric value corresponds with higher importance for that input.

```
In [24]: def variable_importance(model, data, labels, input_vars, score_func, num_iters=10):
    preds = model.predict(data)[:, 0]
    score_val = score_func(labels, preds)
    indices = np.arange(data.shape[0])
    imp_scores = np.zeros((len(input_vars), num_iters))
    shuf_data = np.copy(data)
    for n in range(num_iters):
        print(n)
        np.random.shuffle(indices)
        for v, var in enumerate(input_vars):
            print(var)
            shuf_data[:, :, :, v] = shuf_data[indices, :, :, v]
            shuf_preds = model.predict(shuf_data)[:, 0]
            imp_scores[v, n] = score_func(labels, shuf_preds)
            shuf_data[:, :, :, v] = data[:, :, :, v]
    return score_val - imp_scores
```

Code from DJ Gagne (CISL):

https://github.com/djgagne/swirlnet/blob/master/notebooks/deep_swirl_tutorial.ipynb

```
# Define function for feature importance
def permutation_feature_importance(input_data, output_data, model,
                                     score=mse_preds):
    model_preds = model.predict(input_data)
    original_error = score(output_data, model_preds)
    permuted_scores = np.zeros(input_data.shape[1])
    permuted_data = np.copy(input_data)
    permuted_indices = np.arange(input_data.shape[0])
    for c in range(input_data.shape[1]):
        np.random.shuffle(permuted_indices)
        permuted_data[:, c] = input_data[permuted_indices, c]
        permuted_preds = model.predict(permuted_data)
        permuted_scores[c] = score(output_data, permuted_preds)
        permuted_data[:, c] = input_data[:, c]
    return original_error, permuted_scores

# Calculate feature importance
gpp_original, gpp_permuted = permutation_feature_importance(inputdata,
                                                               outputdata_GPP, model_GPP)
lhf_original, lhf_permuted = permutation_feature_importance(inputdata,
                                                               outputdata_LHF, model_LHF)

# Plot importance
# smaller bar is more important
plt.figure(figsize=(12, 5))
plt.subplots_adjust(wspace=0.4)
plt.subplot(1, 2, 1)
plt.barh(np.arange(gpp_permuted.size), gpp_permuted)
plt.barh(gpp_permuted.size, gpp_original)
plt.yticks(np.arange(gpp_permuted.size + 1),
           in_vars + ["GPP Original"])
plt.title("GPP Importances")
plt.subplot(1, 2, 2)
plt.barh(np.arange(lhf_permuted.size), lhf_permuted)
plt.barh(lhf_permuted.size, lhf_original)
plt.yticks(np.arange(lhf_permuted.size + 1),
           in_vars + ["LHF Original"])
plt.title("LHF Importances")
plt.show()
```

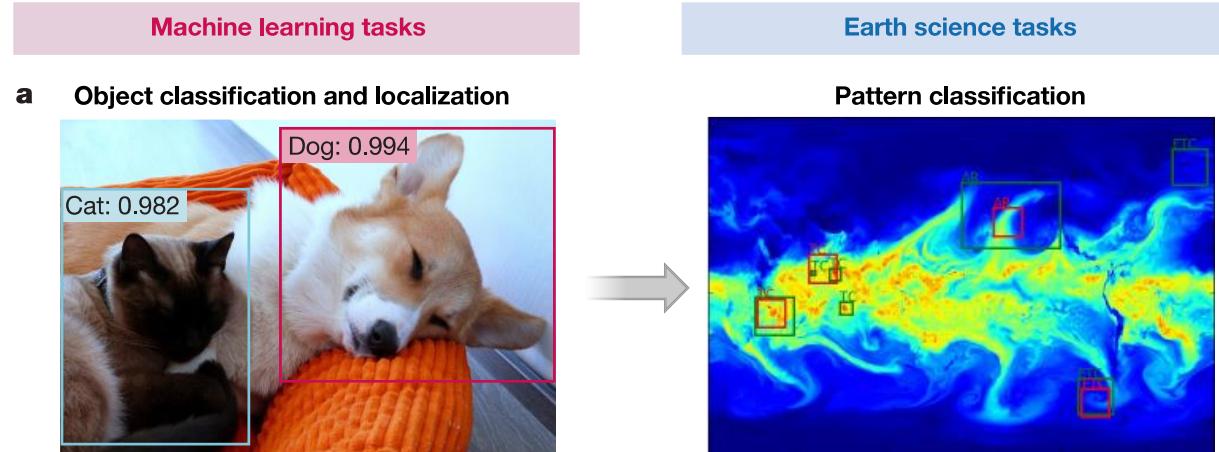
https://github.com/katiedagon/CLM5_ParameterUncertainty

Image Detection for Weather Features

Extreme precipitation has significant impacts, and originates from many different sources.



We can leverage machine learning for automated feature detection.



Reichstein et al. (2019)

- Apply **machine learning-based detection algorithm** to automate the classification of weather features such as **fronts** and **atmospheric rivers (ARs)**.
 - Study the **causes of extreme precipitation** by associating features with extreme precipitation events.



Office of
Science

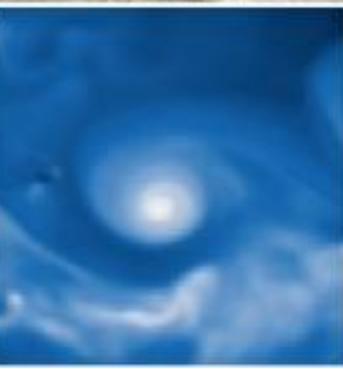


The NCAR logo consists of the letters "NCAR" in a bold, black, sans-serif font. To the left of the text is a graphic element: a blue horizontal bar with a thin orange line running through it.

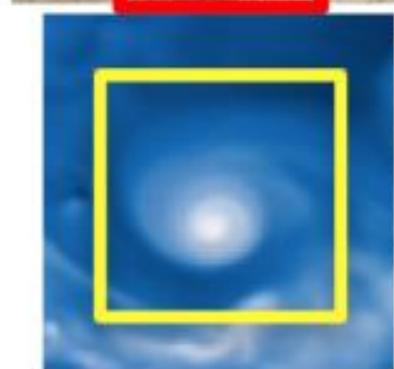
Computer Vision <> Climate Science



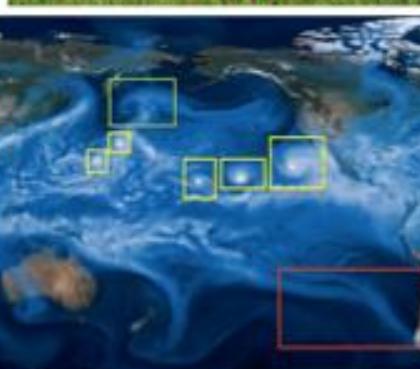
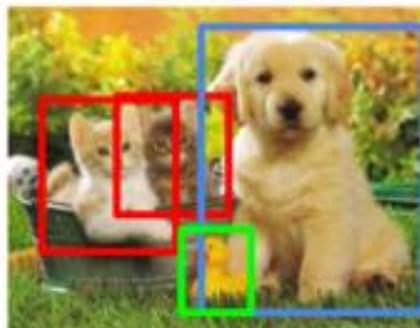
Classification



Classification + Localization



Object Detection



Instance Segmentation



Key Differences: Multi-variate; Different & multiple spatio-temporal scales; Double precision floating point; Underlying statistics are different.



U.S. DEPARTMENT OF
ENERGY

Office of
Science

- 11 -

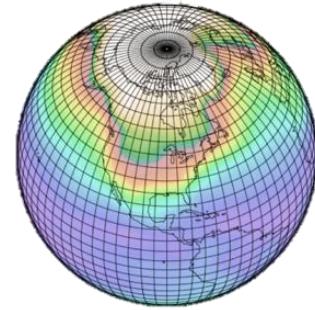


Slide from Karthik
Kashinath

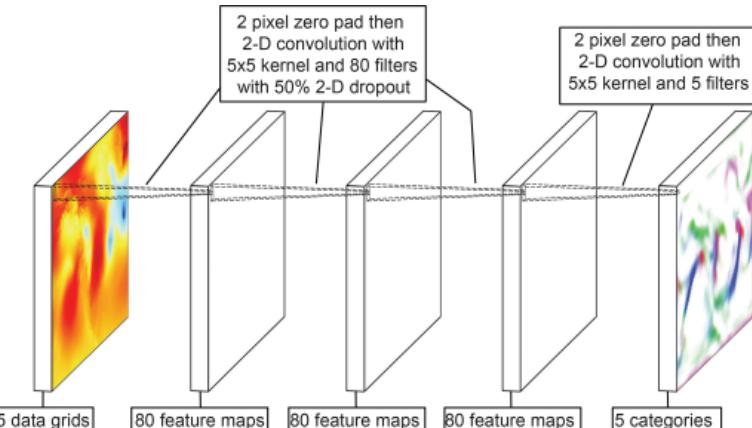
Machine Learning for Detection of Fronts



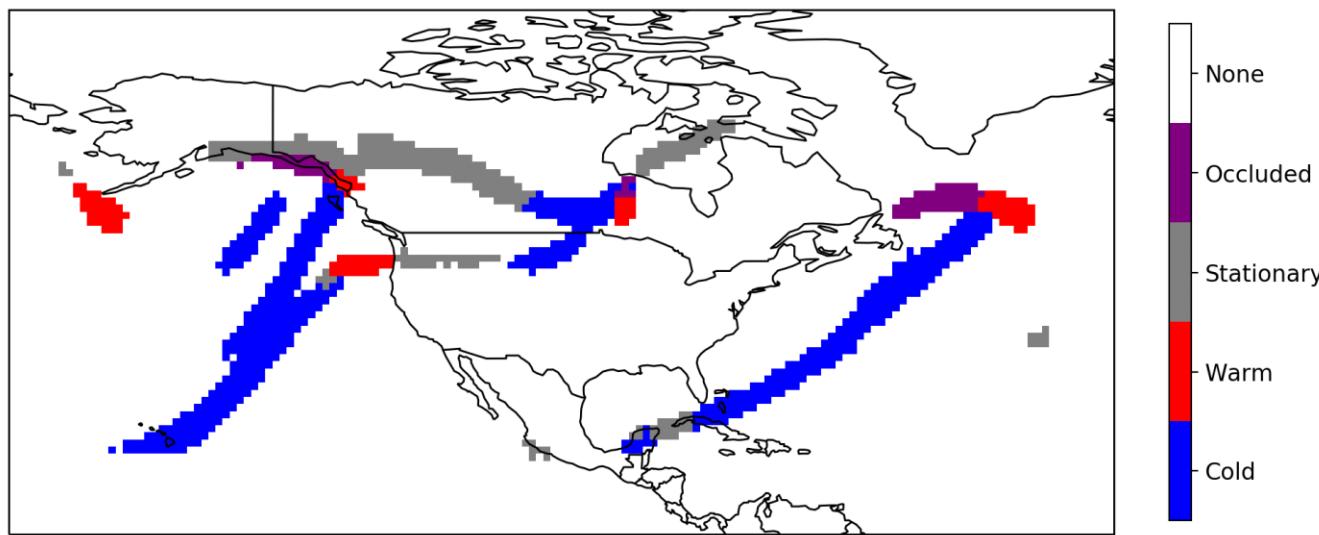
Using the pretrained DL-FRONT algorithm applied to climate model output to detect weather fronts



Specifically, fully coupled Community Earth System Model (CESM) simulations at high spatial and temporal resolution.



For details on DL-FRONT, see: Biard and Kunkel (2019)



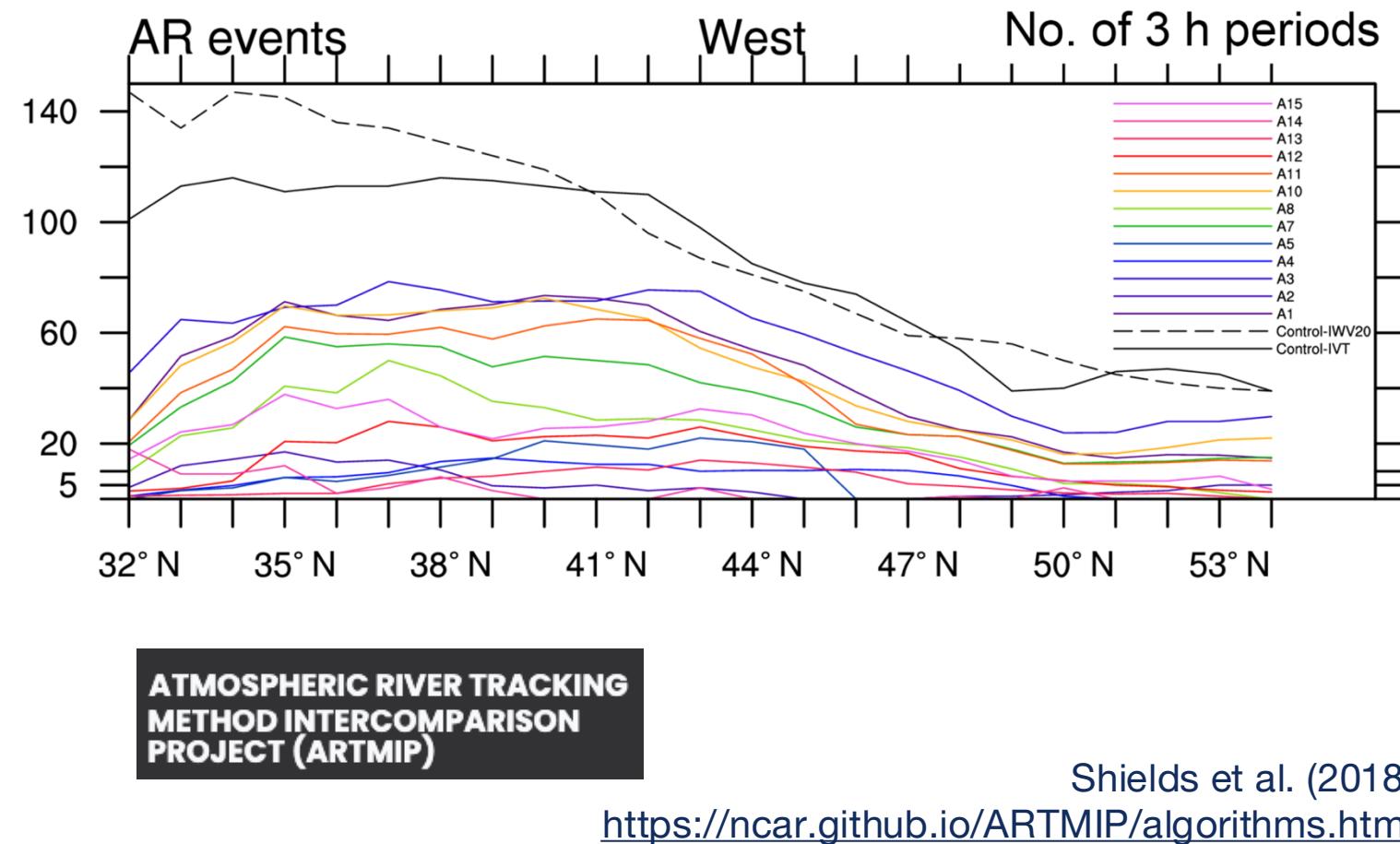
Input fields: surface temperature, sea level pressure, specific humidity and u/v winds
Output fields: 4 frontal categories (cold, warm, occluded, stationary) and none type
Simulation years: Historical climate simulation, 2000-2015
Animation shows 3hrly, 1deg output; Jan 2000.

With: John Truesdale (NCAR), Jim Biard (ClimateAI), Ken Kunkel (NC State), Jerry Meehl (NCAR), Maria Molina (NCAR/UMD)

Dagon et al. (2022)

ML vs. Traditional Detection

- Traditional detection methods use specific thresholds of physical quantities (e.g., “heuristics”)
- For ARs specifically, there is a lot of uncertainty across detection methods (see ARTMIP papers)
- Using ML allows for threshold-free detection, which can aid in the objectivity and computational expense of these algorithms
- We can also test on different data sources and climate states

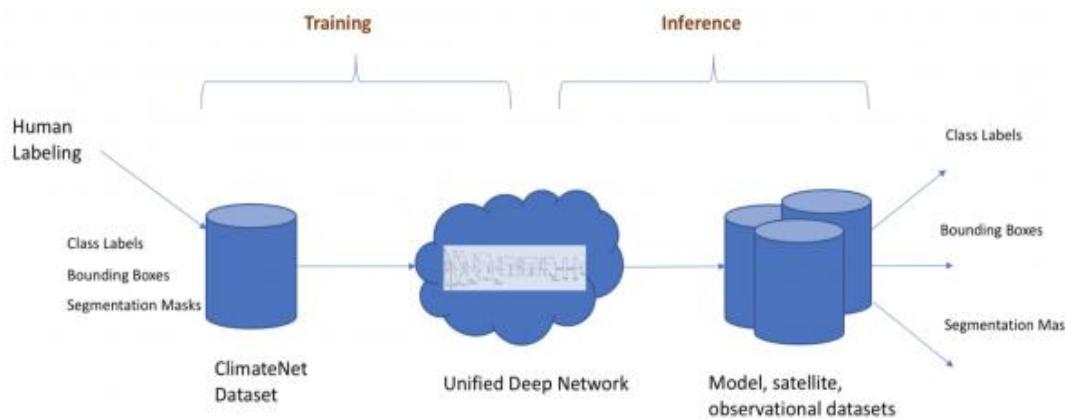


➤ But where do we get the **training data** for this supervised learning application?

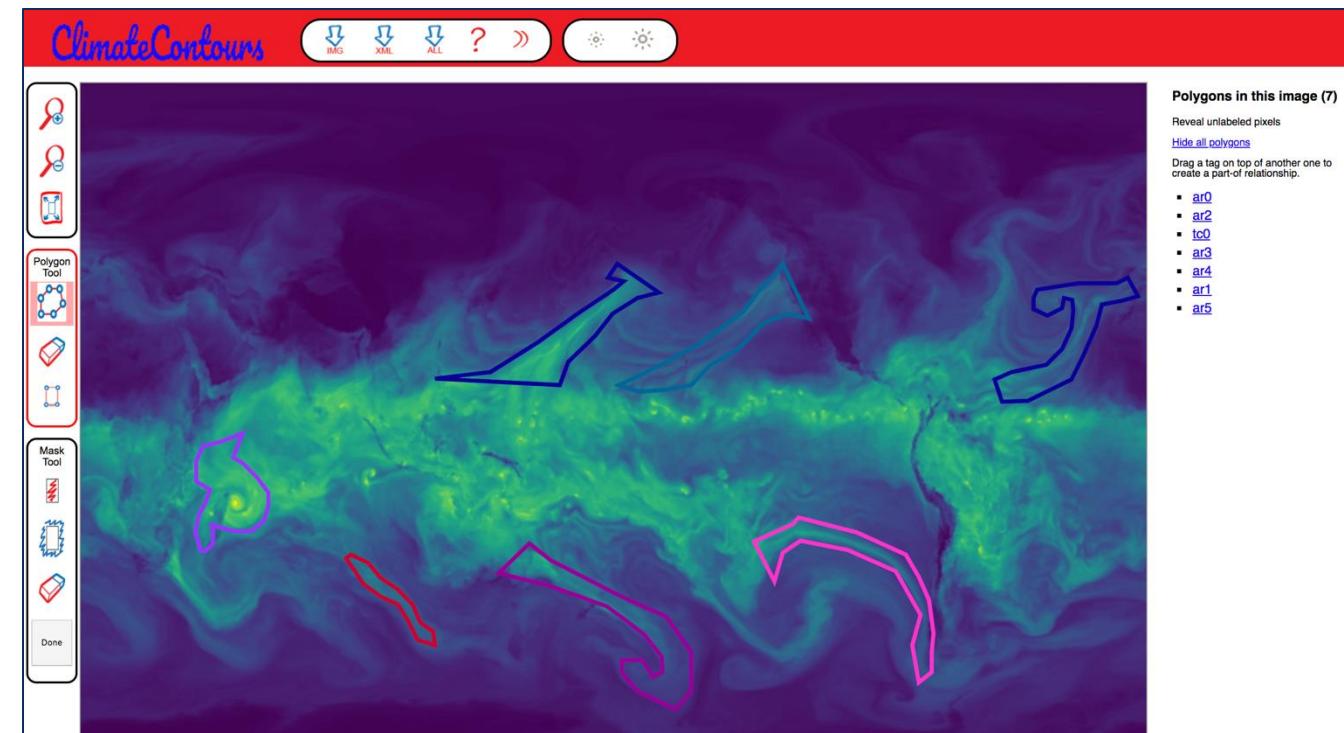
Machine Learning for Detection of ARs and TCs

ClimateNet: a community-sourced expert-labeled dataset to improve and accelerate machine learning applications in weather and climate

- Focus on detecting atmospheric rivers (ARs) and tropical cyclones (TCs).



For details on ClimateNet, see:
Prabhat et al. (2021)



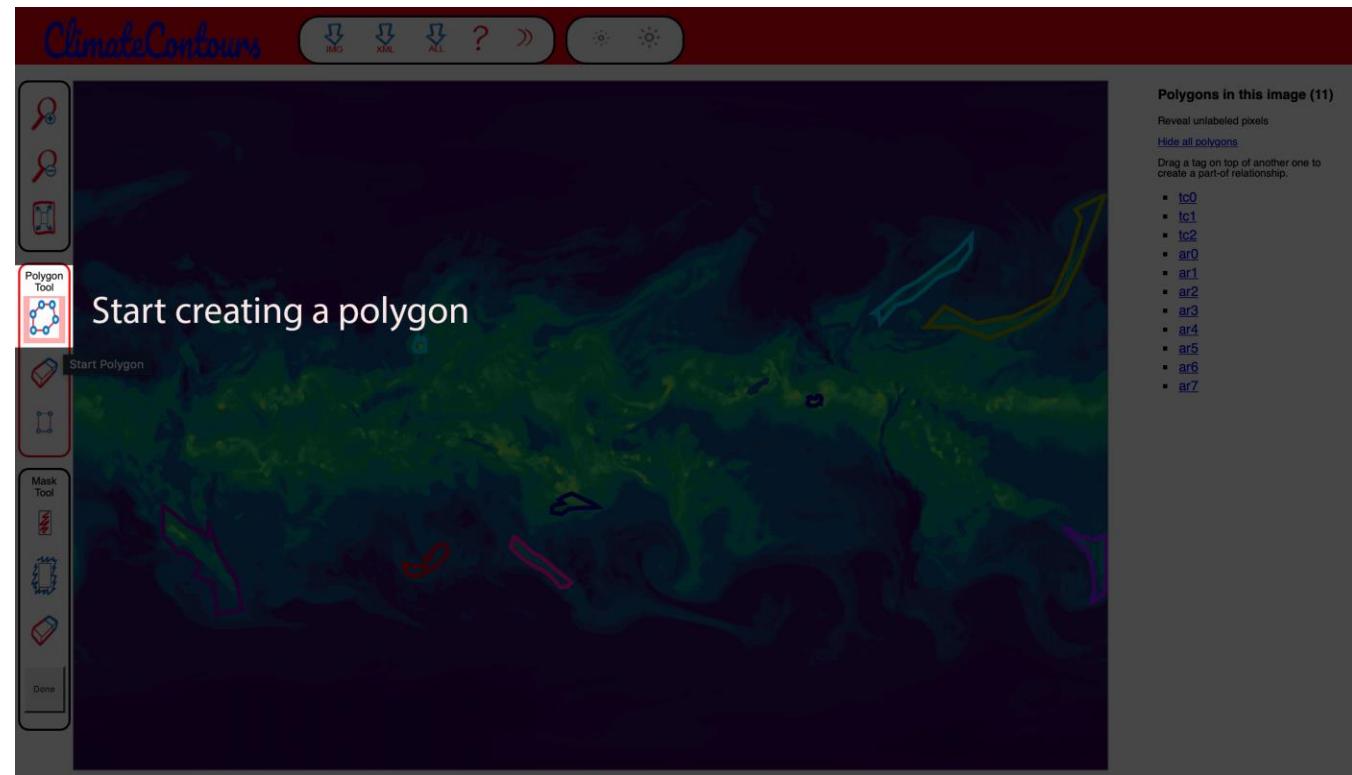
Online Labeling Tool

Tool samples a **random time point** from climate model output.

User **toggles between output fields** (e.g., water vapor, pressure, winds) to categorize events.

User draws boundaries around **atmospheric rivers or tropical cyclones**, repeats for all events in the image.

User submits responses with **confidence level** for each event (low, medium, high).



Labeling Campaigns

ClimateNet: a community-sourced expert-labeled dataset to improve and accelerate Machine Learning applications in weather and climate

Pattern recognition using Machine Learning is data hungry and needs plenty of reliable *labeled* data. Climate science is data rich but reliable labeled data is sparse!

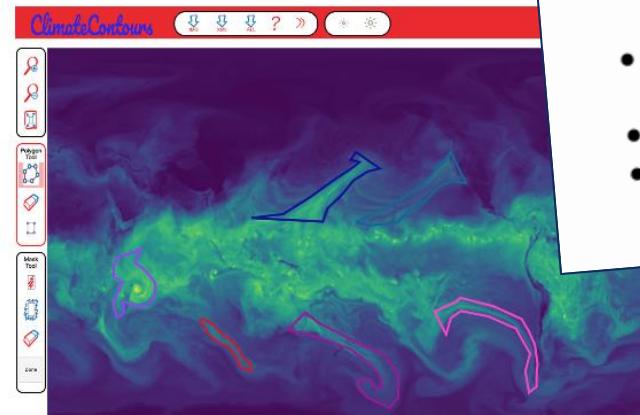
Come join us in a community effort to provide high quality labeled data for a range of important event detection and tracking problems, such as tropical cyclones and atmospheric rivers

When: Nov 20, 10am-12pm MT

Where: Damon Room at NCAR

Remote access: <https://lbnl.zoom.us/j/5104862667>

RSVP: Christine Shields <shields@ucar.edu>, Katie Dagon <kdagon@ucar.edu>, Karthik Kashinath <kkashinath@lbl.gov>



<https://www.nersc.gov/research-and-development/data-analytics/big-data-center/climatenet>

DO's

- Look for **long, narrow** columns transporting water from the equator to the mid-latitudes
- A good guide (not set in stone at all)/ rule of thumb is areas of high IVT (~750 kg/m/s) or IWV (20-25 mm)
- Longer than it is wide: ~1500 km long, less than 1000 km wide. There should be a roughly 2:1 ratio (at least).
- I usually find it useful to check both IWV and IVT when labelling atmospheric rivers
- At any given time, there can be anywhere from ~6~12 (loose numbers) atmospheric rivers in the world

DON'Ts

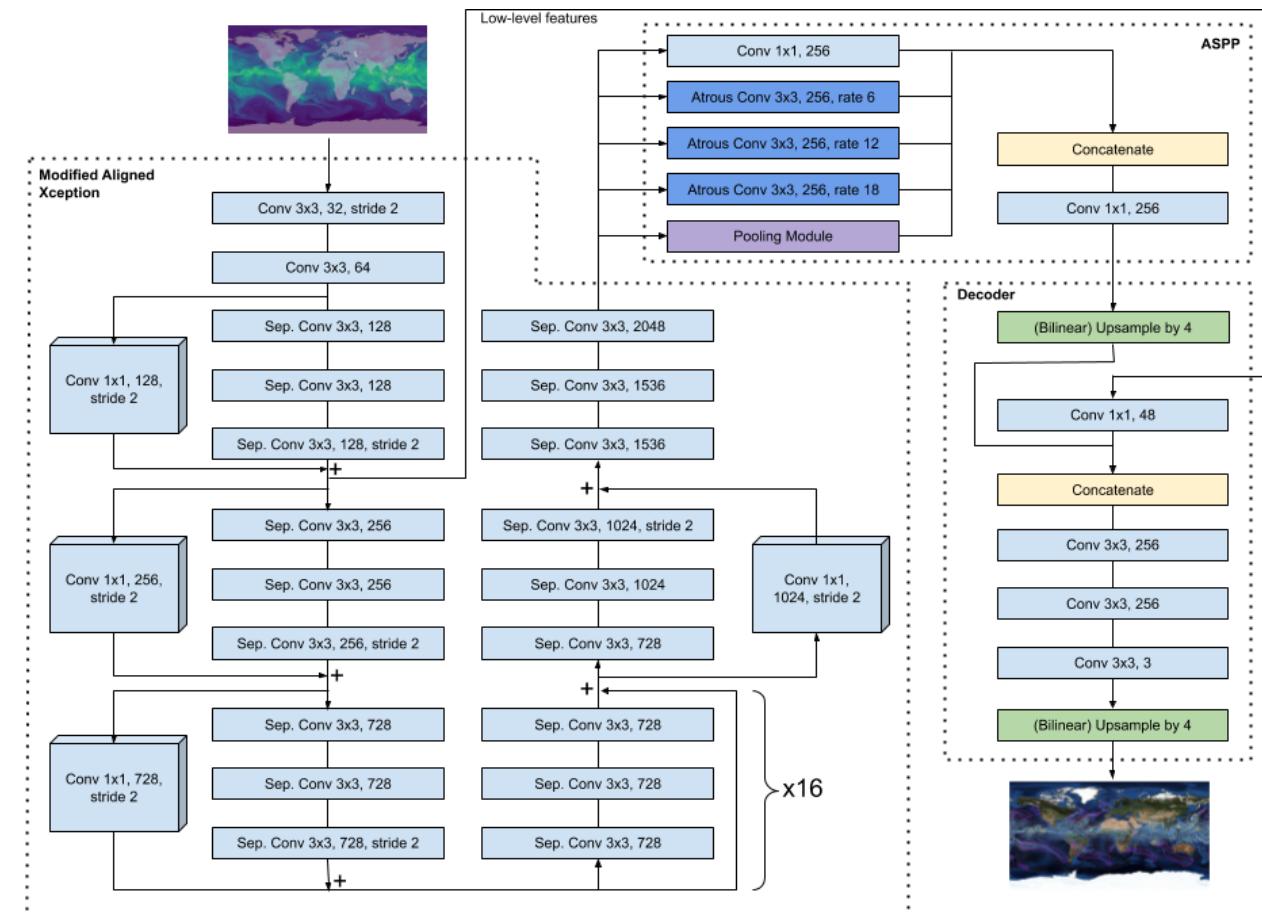
- Use vorticity in AR detection/ identification
- Label areas near the equator as areas with high IVT
- Label small blobs as atmospheric rivers

After multiple labeling campaigns (LBNL, UC Berkeley, NCAR, Scripps/UCSD, 2019 ARTMIP Workshop, 2019 Climate Informatics) with ~80 experts participating, quality control process, **459 images were finalized for training.**



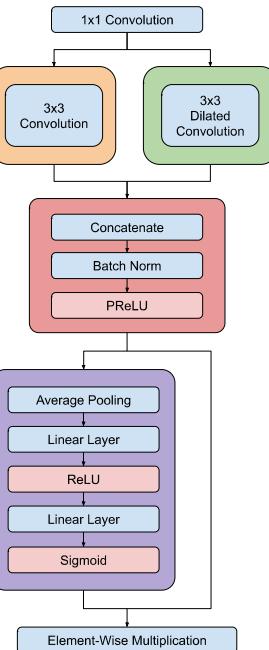
Trained Models: ClimateNet and CGNet

ClimateNet: based on DeepLabv3+ network architecture



Prabhat et al. (2021), Chen et al. (2018)

CGNet: a light-weight “context-guided” network for image segmentation



- Fast and scalable
- Fewer trainable parameters
- Introduces spatio-temporal tracking



<https://github.com/andregraubner/ClimateNet>

Figure 1: A CGBlock integrates local features and surrounding features into a joint feature. It then refines the representation using global context.

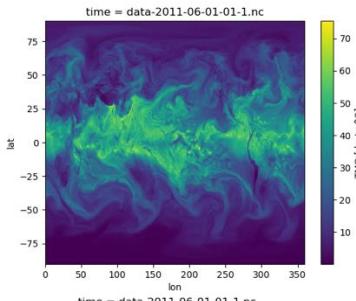
Kapp-Schwoerer et al. (2020), Wu et al. (2019)

Input: maps of climate variables

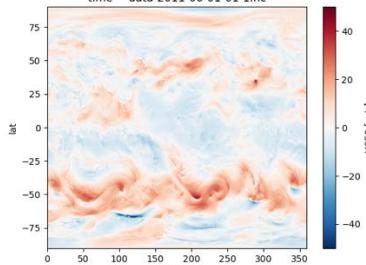
Training CGNet



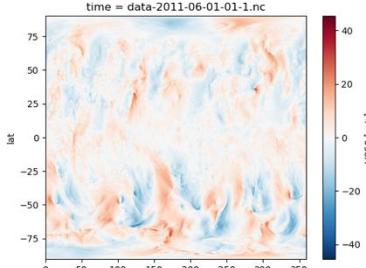
Total column water vapor



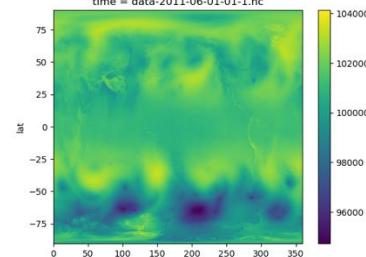
850 hPa zonal wind



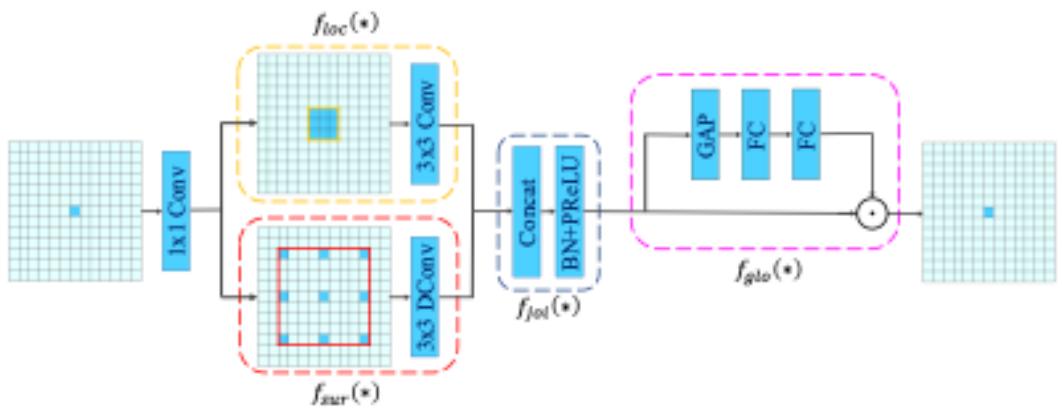
850 hPa meridional wind



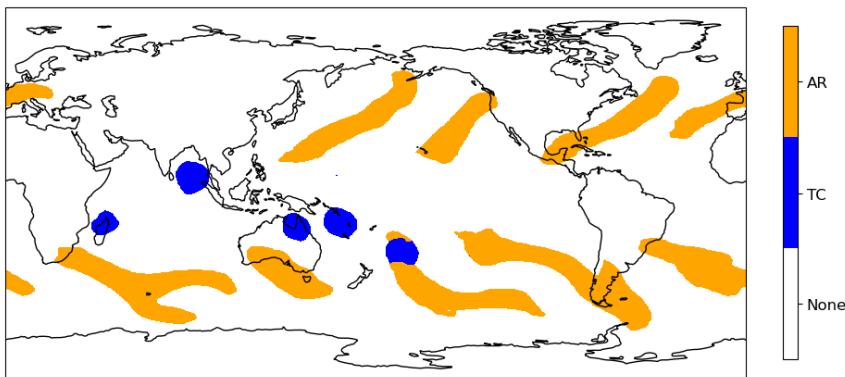
Sea level pressure



CNN (CGNet)



Output: human labels of ARs and TCs



Learning the mapping from climate state to detected feature

With: Lukas Kapp-Schwoerer (Google), Andre Graubner, Sol Kim (Univ. Chicago), Karthik Kashinath (NVIDIA)

Not Reinventing the Wheel

andregraubner / ClimateNet

Code Issues 4 Pull requests 2 Actions Projects Security Insights

Files

main + Q

Go to file t

climatenet

LICENSE

README.md

conda_env.yml

config.json

example.py

ClimateNet / example.py

andregraubner Updated example.py to move epochs to config

Code Blame 31 lines (23 loc) · 1022 Bytes

```
1 from climatenet.utils.data import ClimateDatasetLabeled, ClimateDataset
2 from climatenet.models import CGNet
3 from climatenet.utils.utils import Config
4 from climatenet.track_events import track_events
5 from climatenet.analyze_events import analyze_events
6 from climatenet.visualize_events import visualize_events
7
8 from os import path
9
10 config = Config('config.json')
11 cgnnet = CGNet(config)
12
13 train_path = 'PATH_TO_TRAINING_SET'
14 inference_path = 'PATH_TO_INFERENCE_SET'
15
16 train = ClimateDatasetLabeled(path.join(train_path, 'train'), config)
17 test = ClimateDatasetLabeled(path.join(train_path, 'test'), config)
18 inference = ClimateDataset(inference_path, config)
19
20 cgnnet.train(train)
21 cgnnet.evaluate(test)
22
23 cgnnet.save_model('trained_cgnnet')
24 # use a saved model with
25 # cgnnet.load_model('trained_cgnnet')
26
27 class_masks = cgnnet.predict(inference) # masks with 1==TC, 2==AR
28 event_masks = track_events(class_masks) # masks with event IDs
29
30 analyze_events(event_masks, class_masks, 'results/')
31 visualize_events(event_masks, inference, 'pngs/')
```

Training CGNet

Purpose:

The purpose of this notebook is to train CGNet for machine learning detection of atmospheric rivers and tropical cyclones.
See ClimateNet repo here: <https://github.com/andregraubner/ClimateNet>

Authors/Contributors:

- Teagan King
- John Truesdale
- Katie Dagon

Import libraries

```
[1]: import os
import sys
import json
import numpy as np
import matplotlib.pyplot as plt

sys.path.append("/glade/work/kdagon/ClimateNet") # append path to ClimateNet repo
from climatenet.utils.data import ClimateDatasetLabeled, ClimateDataset
from climatenet.models import CGNet
from climatenet.utils.utils import Config
from climatenet.track_events import track_events
from climatenet.analyze_events import analyze_events
from climatenet.visualize_events import visualize_events

from os import path
```

<https://github.com/andregraubner/ClimateNet>

<https://github.com/katiedagon/ML-extremes>

Environment & Resources

The screenshot shows a GitHub repository interface for 'ClimateNet'. The repository owner is 'andregraubner'. The 'Code' tab is selected. The 'Files' section on the left shows files like 'main', 'LICENSE', 'README.md', 'conda_env.yml' (which is highlighted), 'config.json', and 'example.py'. The main content area displays the 'conda_env.yml' file content:

```
1 name: climatenet
2 channels:
3   - defaults
4 dependencies:
5   - _libgcc_mutex=0.1=main
6   - _pytorch_select=0.2=gpu_0
7   - blas=1.0=mkl
8   - brotli=0.7.0=py38h27cf23_1003
9   - bzip2=1.0.8=h7b6447c_0
10  - ca-certificates=2020.10.14=0
11  - cartopy=0.18.0=py38hee755e7_0
12  - certifi=2020.12.5=py38h06a4308_0
13  - cffi=1.14.4=py38h261ae71_0
14  - cftime=1.3.0=py38h6323ea4_0
15  - chardet=3.0.4=py38h06a4308_1003
```

NCAR HPC JupyterHub

Resource Selection
Casper PBS Batch

Queue or Reservation (-q)
casper

Project Account (-A)
P06010014

Specify N Nodes (-l select=N)
1

Specify N CPUs per Node (-l ncpus=N)
4

Specify Threads per Process (-l ompthreads=N)
1

Specify MPI processes per Node (-l mpiprocs=N)
1

Specify Memory per Node in GB (-l mem=N)
64

Specify X Number of GPUs / Node (-l ngpus=X)
2

Select GPU Type, X (-l gpu_type=X)
v100

Wall Time HH:MM:SS (24-hour max)
03:00:00

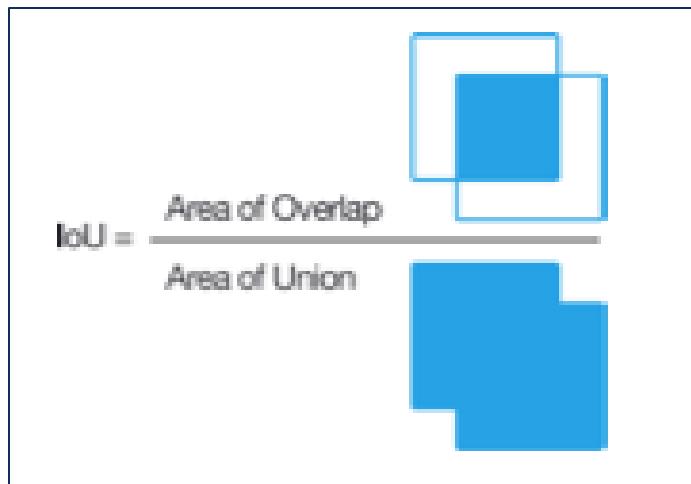
Launch Server

- Some customization to adapt the climatenet conda env to work for our team
- Only works with v100 GPUs (?)

Loss Function & Metrics

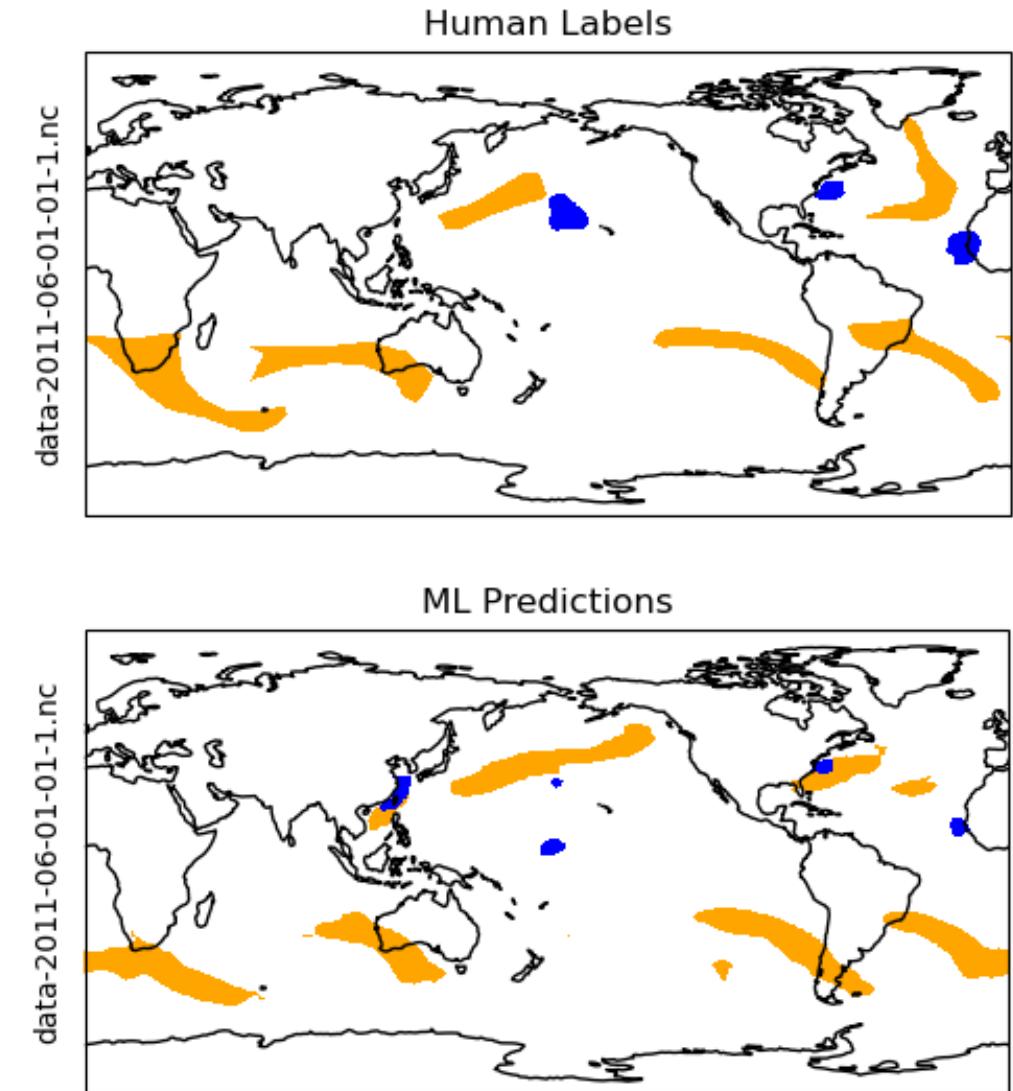
Segmentation problems like these often have an issue called **“class imbalance”** where one of the categories dominates the input data. In this case, it is the “background” class (94% of pixels in ClimateNet).

- One approach is to leverage different loss functions (e.g., weighted cross-entropy, Jaccard loss).



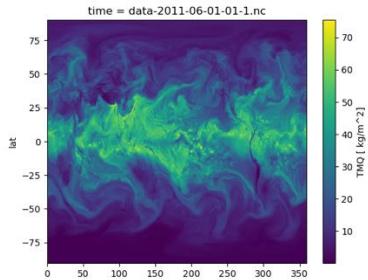
Intersection over Union (IoU) is used to quantify model performance – that is, **how well do the ML generated labels line up with the human labels?**

Prabhat et al. (2021)



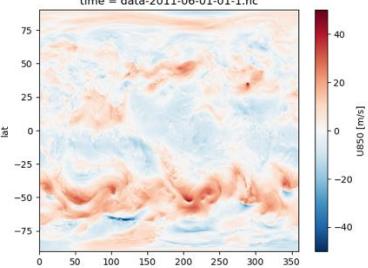
Normalizing the Inputs

Total
column
water
vapor



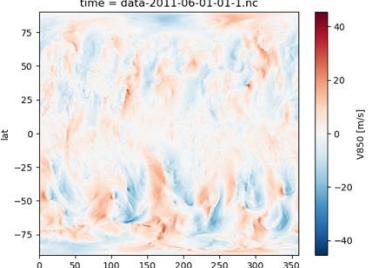
Mean = 19.2 kg/m²

850 hPa
zonal wind



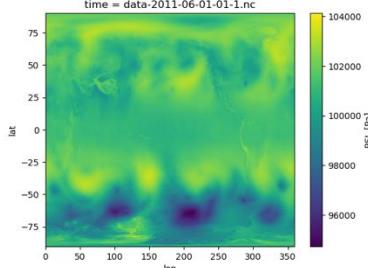
Mean = 1.5 m/s

850 hPa
meridional wind

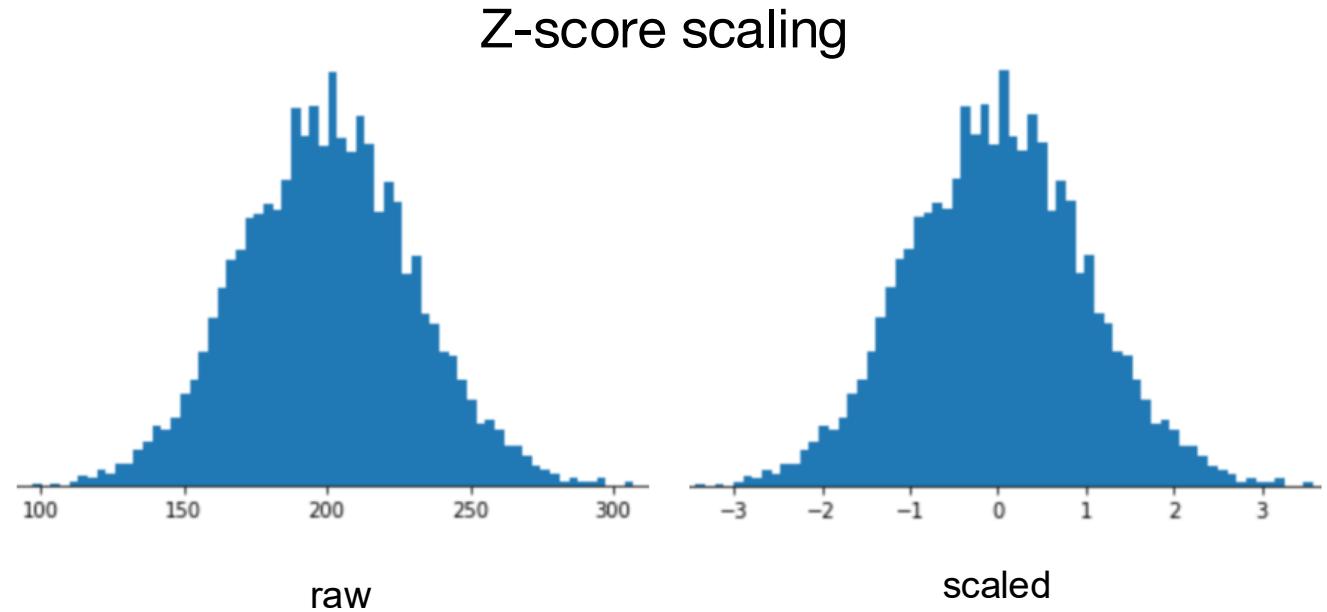


Mean = 0.25 m/s

Sea level
pressure



Mean = 100814 Pa

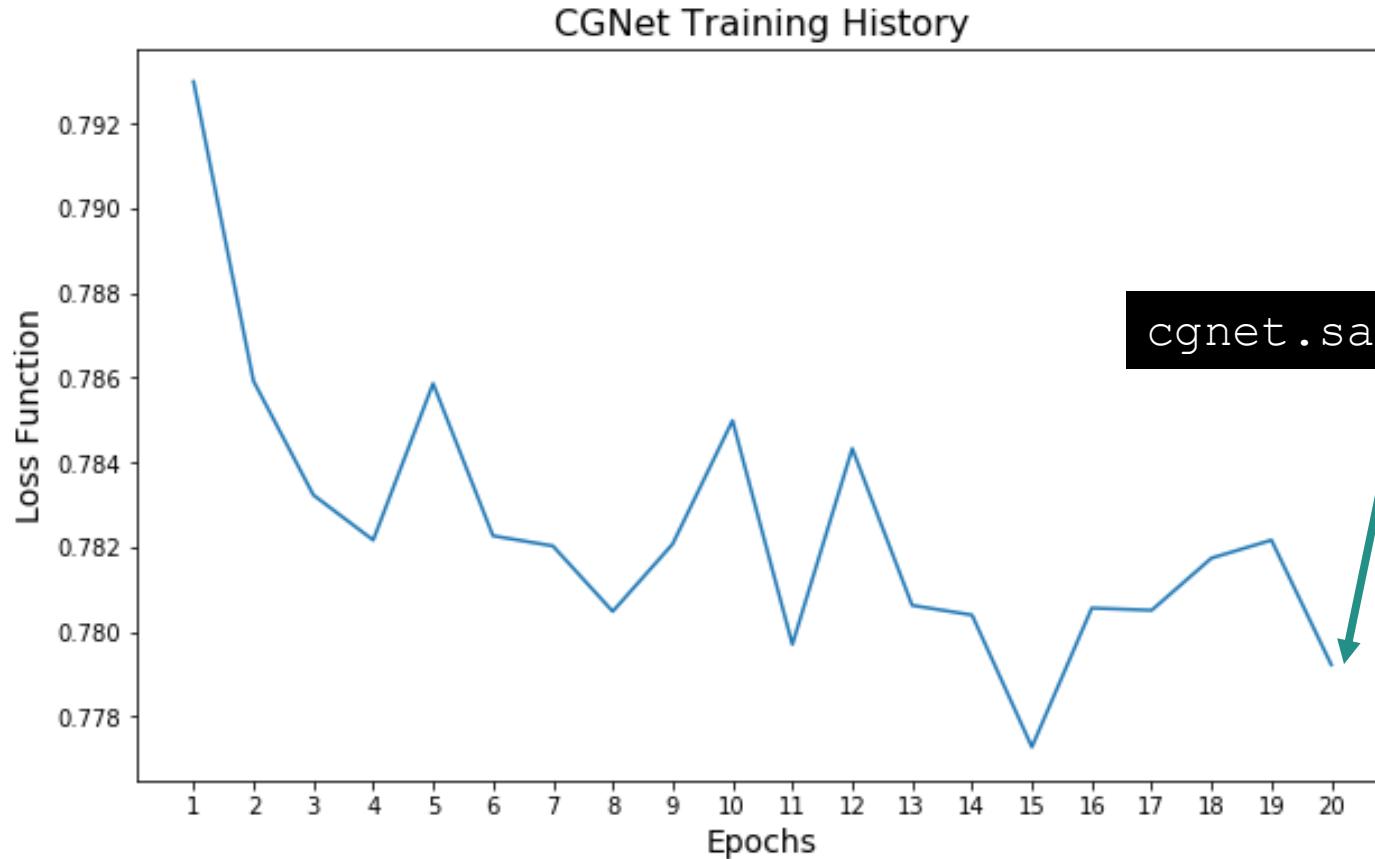


$$x' = (x - \mu) / \sigma$$

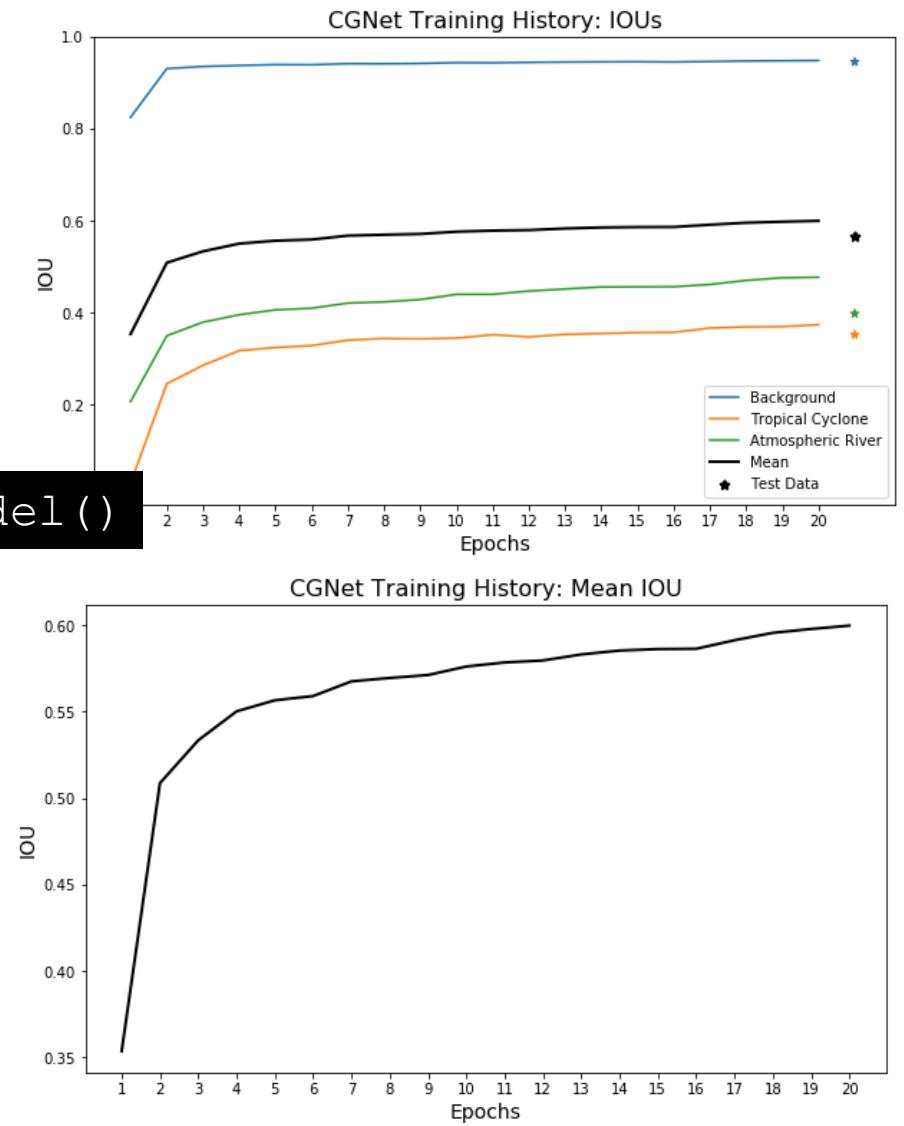
Z-score image from:
<https://developers.google.com/machine-learning/crash-course/numerical-data/normalization>

CGNet Training Statistics

Of the 459 samples, 398 used for training (87%; years 1996-2010) and 61 for testing (13%; years 2011-2013).



Training takes about 1 min per epoch on 2 casper V100 GPUs

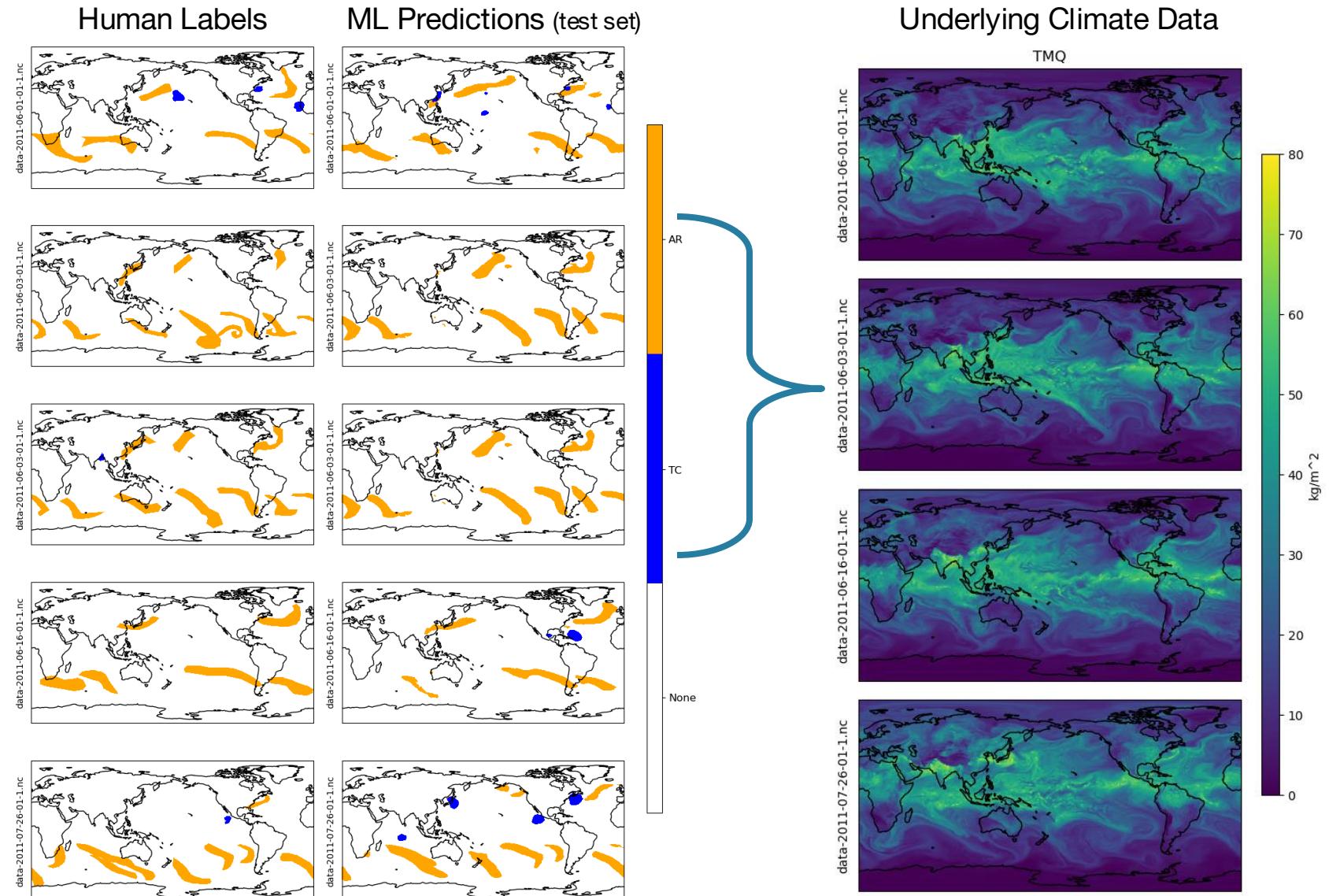
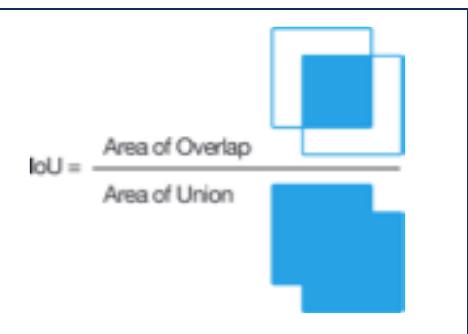


Evaluating CGNet Results: Labels vs. Predictions

How do we evaluate the climate model results?

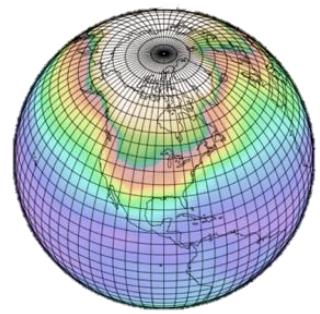
Different expert-labeled maps; different timepoints*

*sometimes two experts label the same timepoint!



Applying the Trained Model to CESM

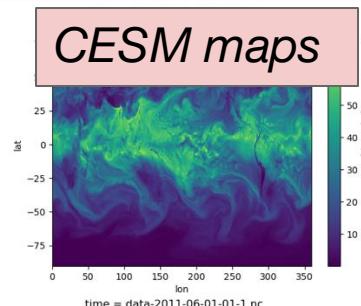
Use the retrained CGNet algorithm applied to climate model output to detect ARs and TCs



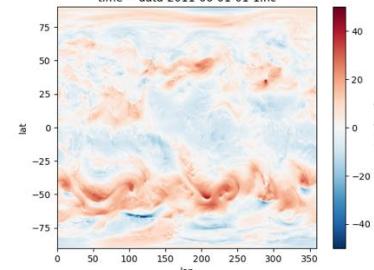
Specifically, fully coupled CESM1.3 simulations at high spatial (0.25deg) and temporal (3hrly) resolution.

Lots of data processing steps went into this work!

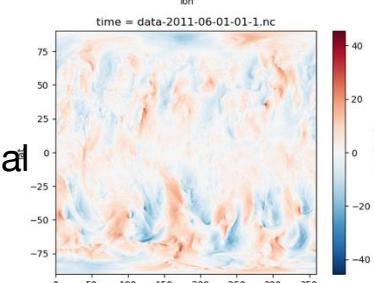
Total column water vapor



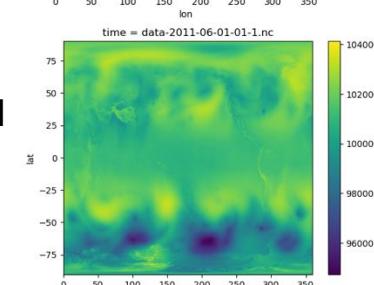
850 hPa zonal wind



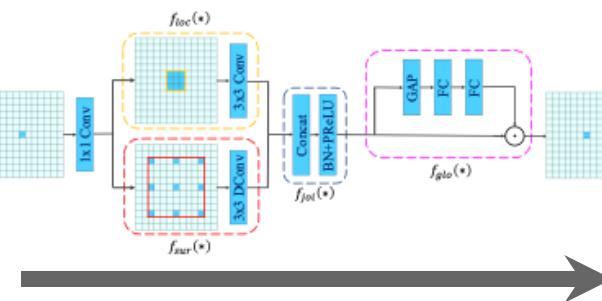
850 hPa meridional wind



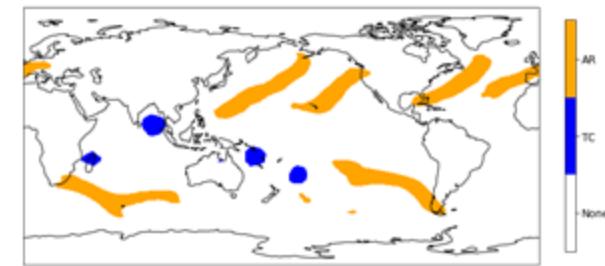
Sea level pressure



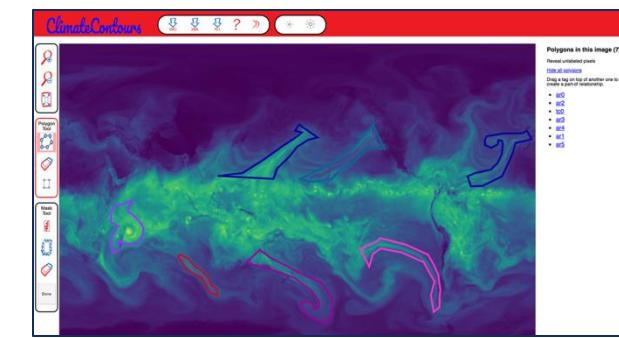
CNN (CGNet)



Labels of ARs and TCs in CESM



Data Processing Workflow



QA/QC
human
labels

Processed data from labeling campaigns (masks & underlying fields)
Training data: labeled timepoints, 1996-2010
Test data: labeled timepoints, 2011-2013
Calculate means/std for normalization



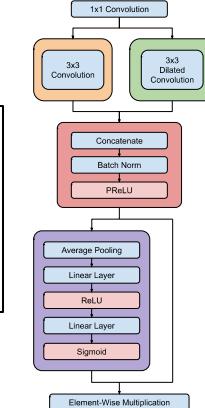
Climate data & HighRes simulations
e.g., for CESM1.3:
Historical 2000-2005
RCP2.6: 2006-2015
RCP8.5: 2086-2100

Processing input fields

- Variable-specific processing, e.g., for CESM1.3:
 - PSL: direct CESM output
 - TMQ: calculate from Q and P
 - U850/V850: interpolate to 850hPa pressure level using PS/U/V, extrapolate high elevation regions
- Remap unstructured mesh to lat/lon grid
- Separate into 3-hourly files

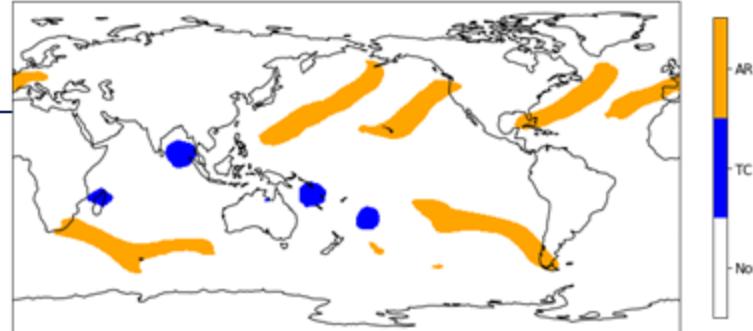
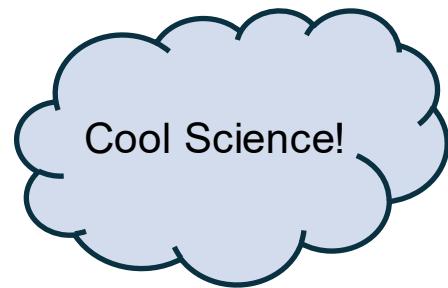
Train CGNet

- 1 input: TMQ
2 inputs: TMQ, V850
4 inputs: TMQ, V850, U850, PSL



Run processed data through CGNet

Generate AR/TC results



NCAR/UCAR Earth System Data Science (ESDS)

UX PROJECT raijin

Analyzing and visualizing CAM-SE output in Python
Overview
We demonstrate a variety of options for analyzing and visualizing output from the Community Atmosphere Model (CAM) with the spectral element (SE) grid in Python. This notebook was developed for the ESDS Collaborative Work Time on Unstructured Grids, which took place on April 17, 2023. A recap of the related CAM-SE discussion can be found [here](#).

<https://ncar.github.io/esds/posts/2023/cam-se-analysis/>

Thanks to **Teagan King** and **John Truesdale** (CGD) for their work on this data processing pipeline



NCAR
Operated by UCAR

K. Dagon

Data Processing Reality

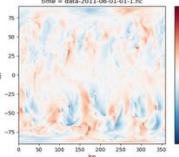
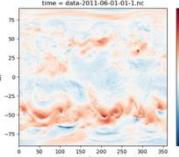
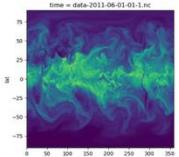


<https://docs.dask.org/en/latest/dashboard.html>

Applying the Trained Model to CESM

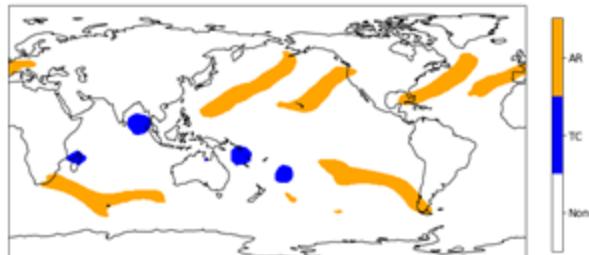
For the CESM historical simulation, 15 years of 3 hourly output translates to **43,800 samples**.

CESM maps



Saved CNN

Labels of ARs and TCs in CESM



Function for creating TC/AR masks using pre-trained model

```
[3]: def cgnnet_load_create_masks(model_path, inference_path, save_dir, analyze=False, visualize=False):
    """Use a pre-trained model to create masks of tropical cyclones (mask value = 1)
    and atmospheric rivers (mask value = 2)

    Function will create NetCDF mask files and save them in the save_directory.

    Parameters:
    -----
    model_path: str
        filepath to pre-trained CGNet model
    inference_path: str
        filepath to inference data
    save_dir: str
        filepath to where the masks will be saved as .nc files
    analyze: bool (optional)
        default is False; if True, will save plots for analyzing events using climatenet.analyze_events().
        Note that this can significantly increase the time to run.
    visualize : bool (optional)
        default is False; if True, will save plots for visualizing events using climatenet.visualize_events().
        Note that this can significantly increase the time to run.

    .....
    # instantiate CGNet with pre-trained config file
    cgnnet = CGNet(model_path=model_path)

    # inference using the pre-trained config file
    inference = ClimateDataset(inference_path, cgnnet.config)
    class_masks = cgnnet.predict(inference) # masks with 1==TC, 2==AR

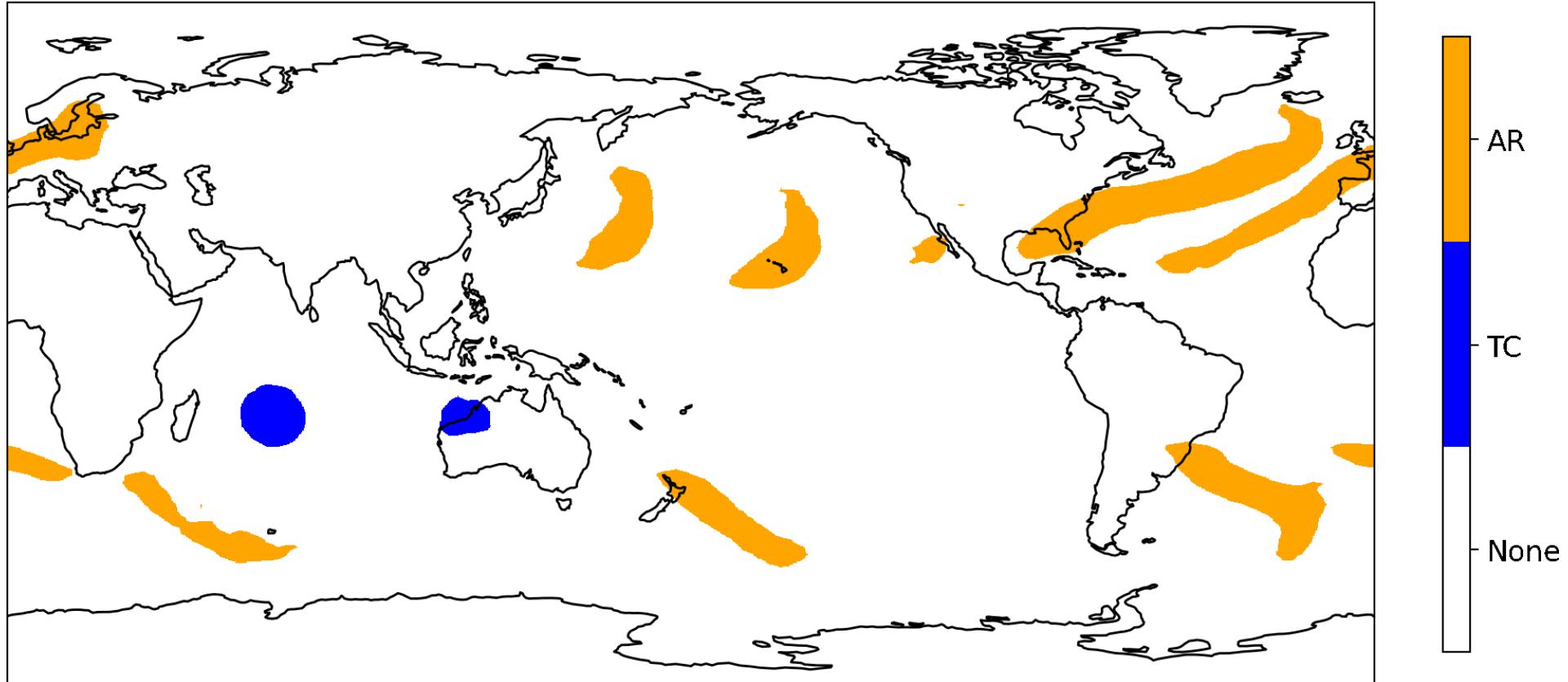
    # create save dir, if needed
    if not os.path.isdir(save_dir):
        os.makedirs(save_dir)
    else:
        print("Warning: might overwrite {}".format(save_dir))

    # save out class masks
    class_masks.name = 'masks'
    class_masks.to_netcdf(save_dir+"/class_masks.nc")
    print("Saved class masks to {}".format(save_dir))
```

Inference takes about 5 min per year on 2 casper V100 GPUs

<https://github.com/katiedagon/ML-extremes>

Applying the Trained Model to CESM



Input fields: TMQ, U850, V850, PSL

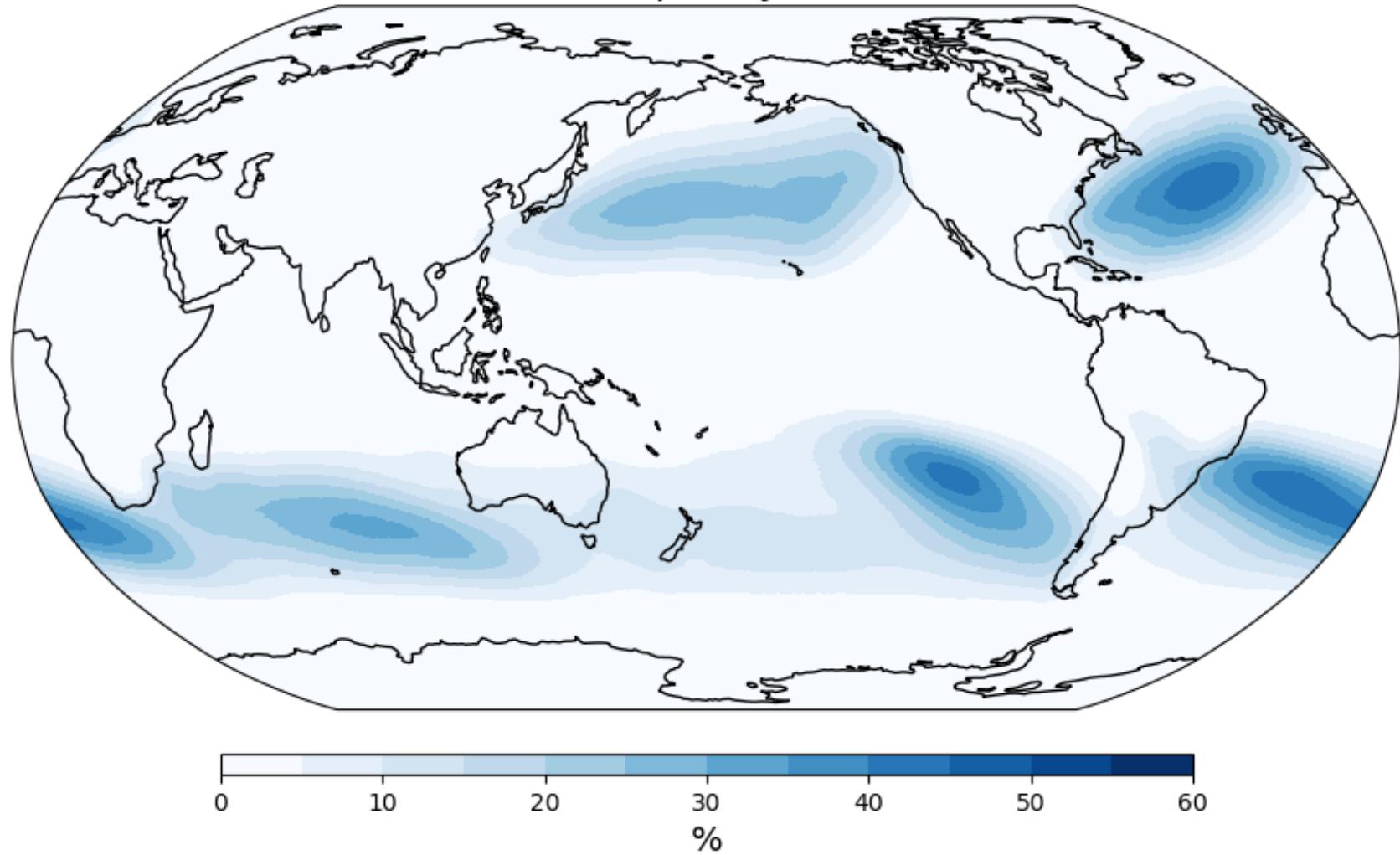
Output fields: AR, TC, and none

Simulation years: CESM1.3 Historical climate simulation, 2000-2015

Animation shows 3hrly, 0.25 deg output; Jan-Feb 2000.

Applying the Trained Model to CESM

CESM AR Frequency, 2000-2015



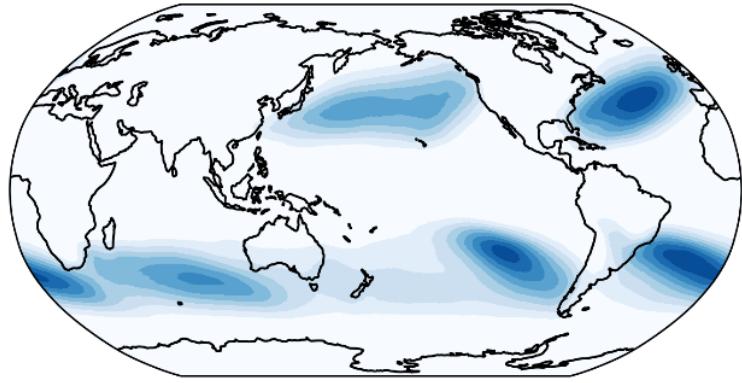
Dagon et al. *in prep*

Sensitivity to Input Fields

CGNet trained on TMQ, U850, V850, PSL

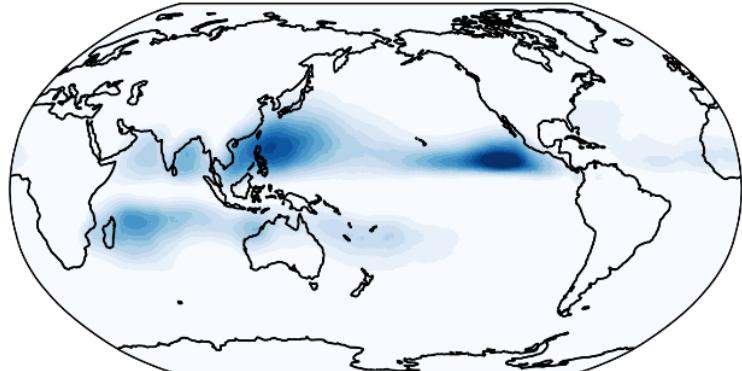
a) ARs: TMQ, U850, V850, PSL

ARs



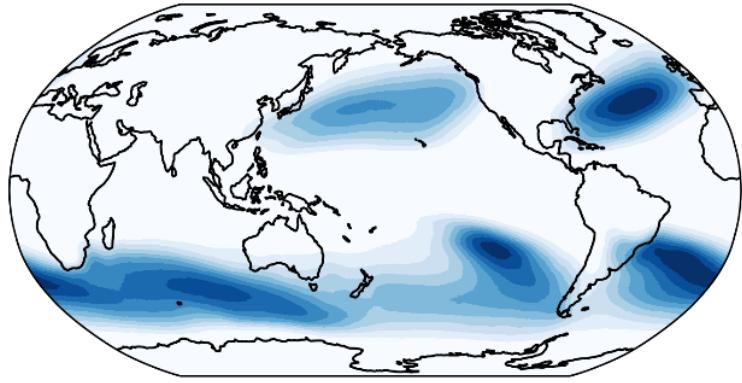
d) TCs: TMQ, U850, V850, PSL

TCs

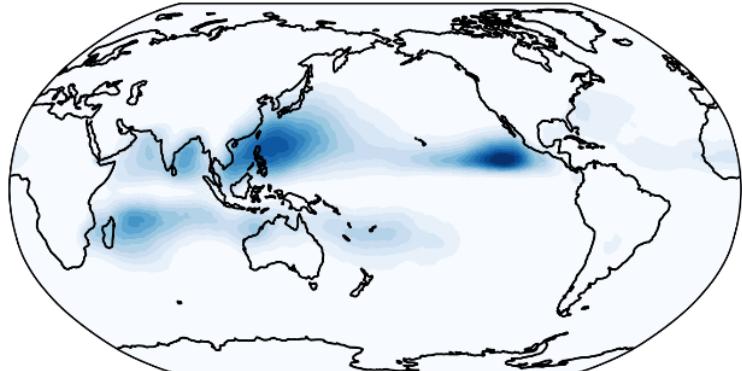


CGNet trained on TMQ, V850

b) ARs: TMQ, V850

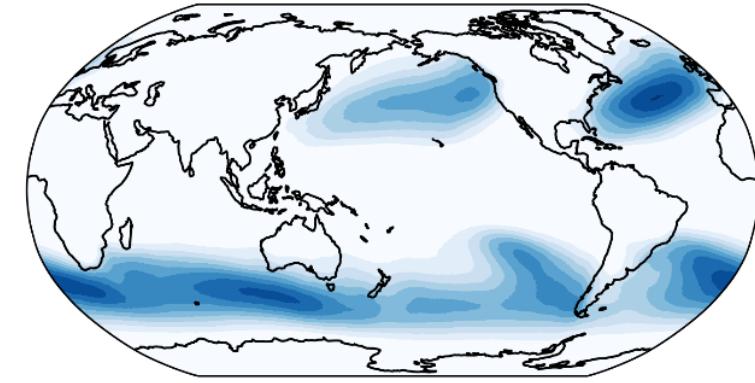


e) TCs: TMQ, V850

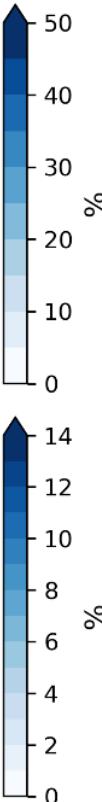
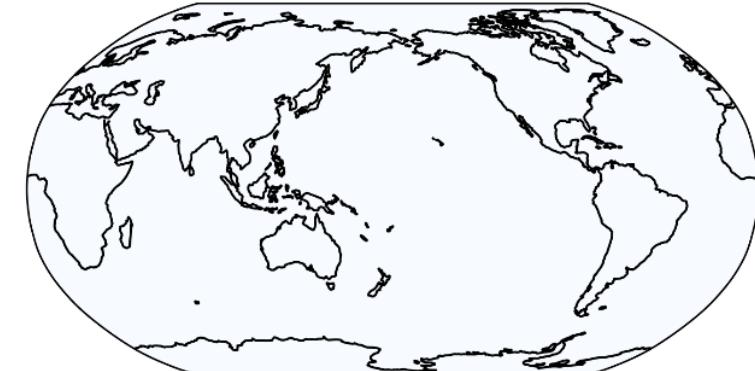


CGNet trained on TMQ

c) ARs: TMQ

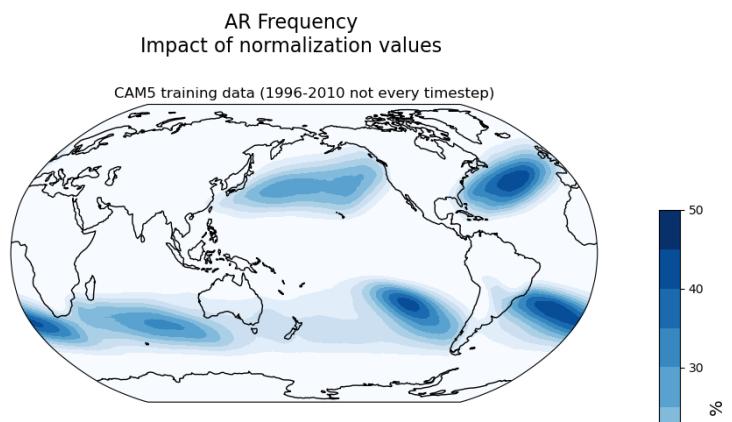


f) TCs: TMQ

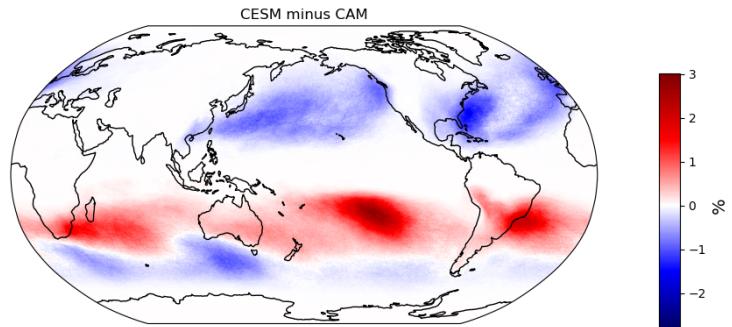
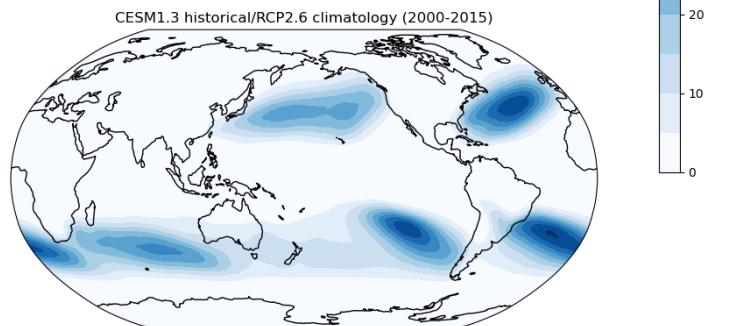


Sensitivity to Normalization

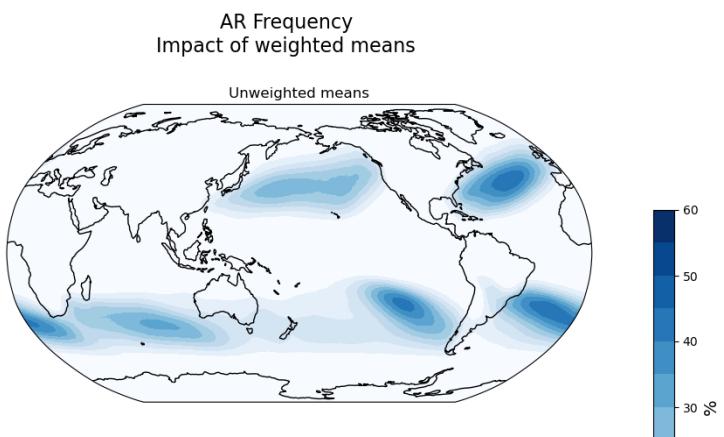
Calculate means/std from the training data (CAM)



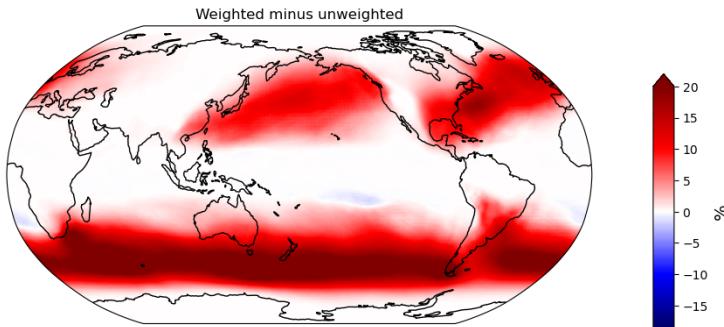
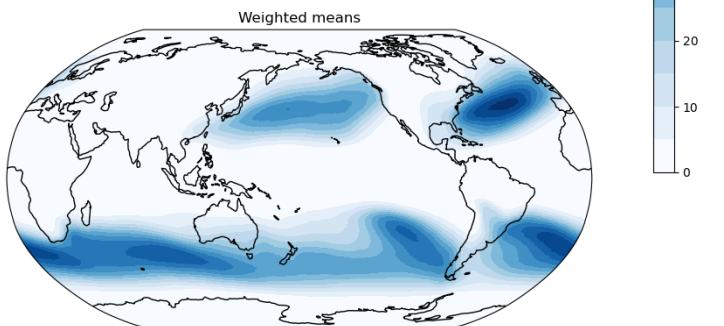
Calculate means/std from the inference data (CESM)



Unweighted means

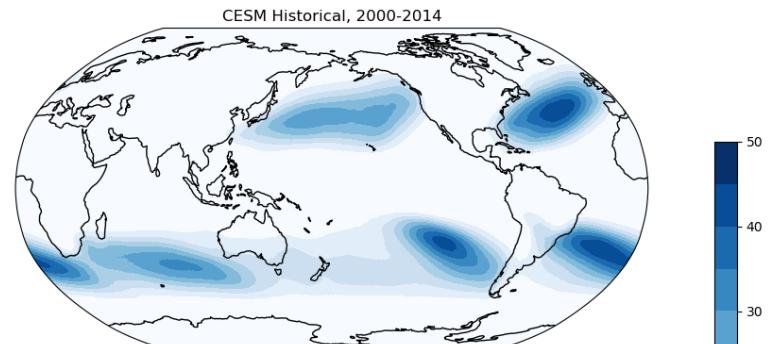


Weighted means (by cosine latitude)

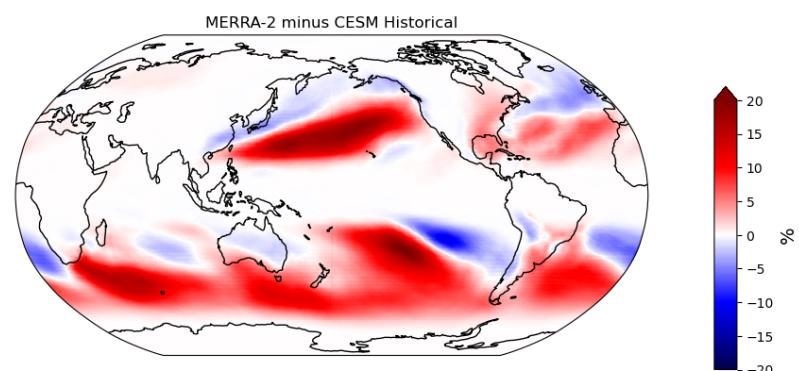
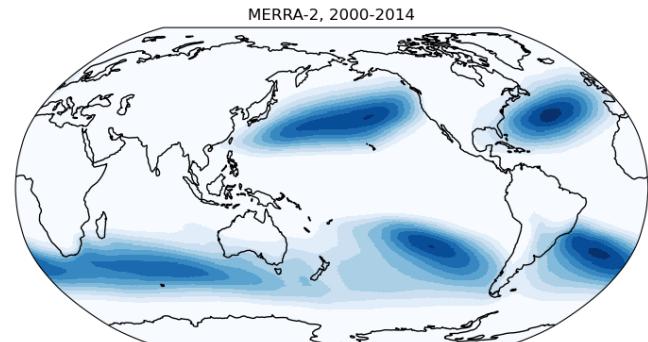


Evaluating CGNet Results: CESM vs. MERRA-2

CESM1.3 with
ne120 grid
(0.25deg resolution)



MERRA-2 native grid
(0.5deg resolution)



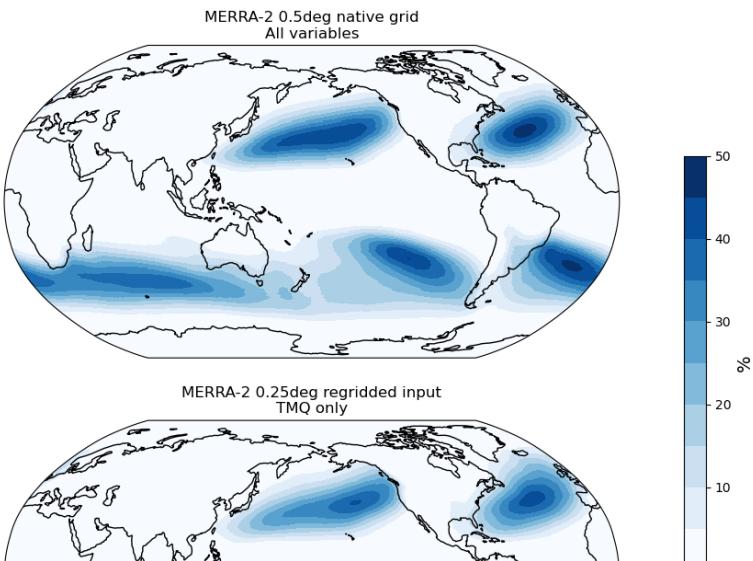
Comparing CESM results to
ARs detected with CGNet in
reanalysis (MERRA-2).

Why are there (predominantly)
more ARs in MERRA-2 than
CESM? Differences between
climate model and reanalysis?
What is the impact of spatial
resolution?

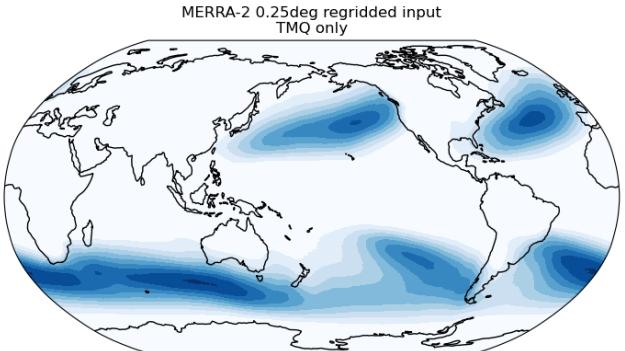
Dagon et al. *in prep*

Sensitivity to Spatial Resolution

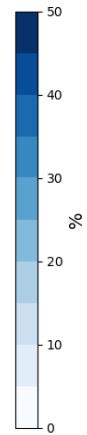
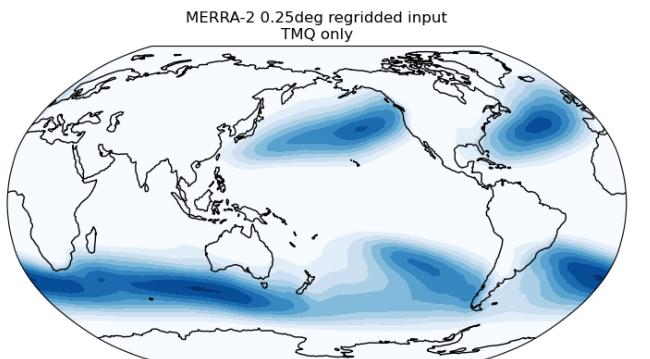
MERRA-2 native grid
(0.5deg resolution)
All input variables



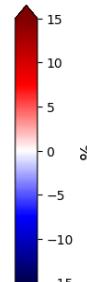
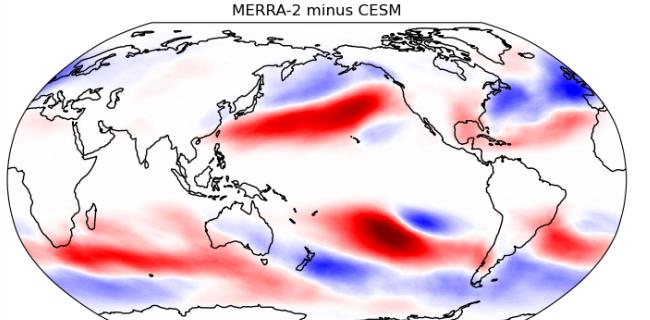
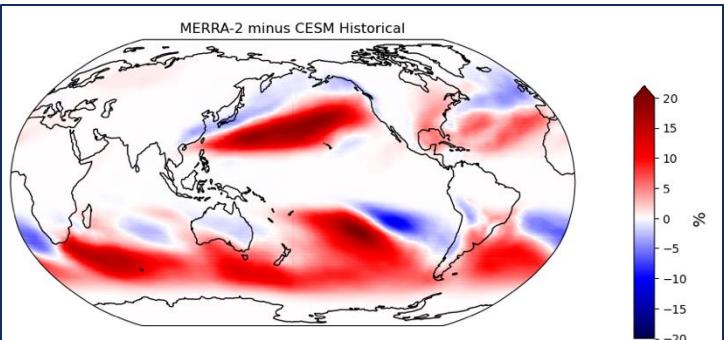
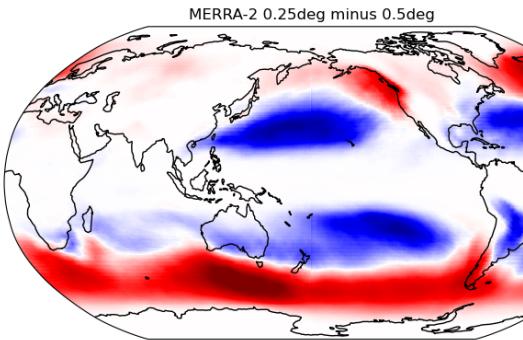
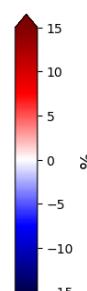
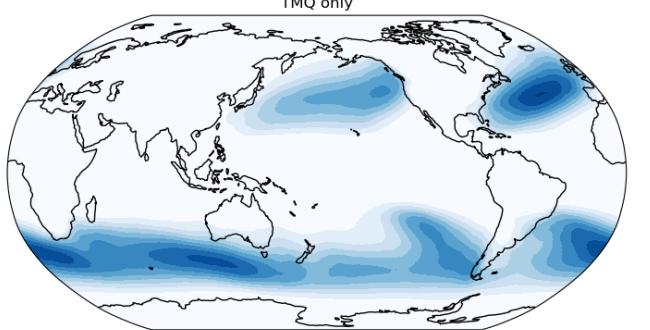
MERRA-2 regressed
(0.25deg resolution)
***Water vapor only**



MERRA-2 regressed
(0.25deg resolution)
Water vapor only

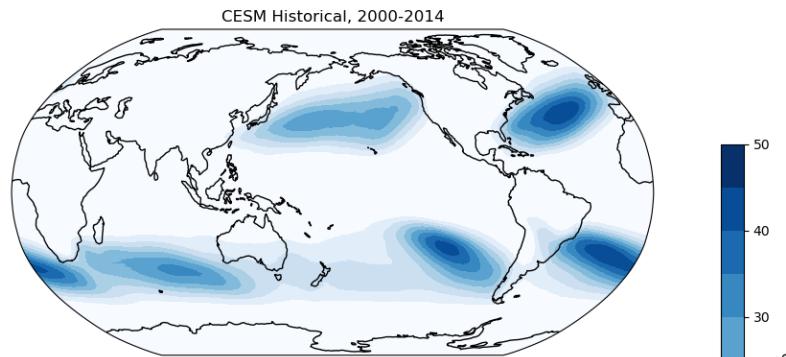


CESM native grid
(0.25deg resolution)
Water vapor only

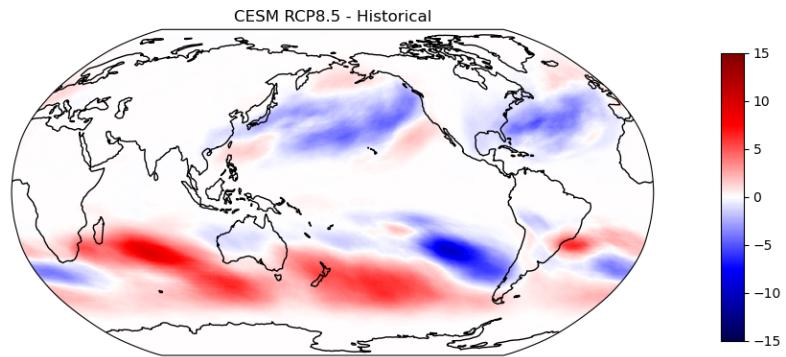
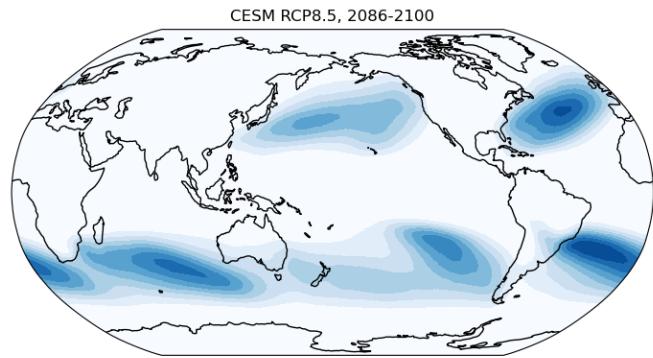


AR Frequency Response to Climate Change

CESM historical,
2000-2014



CESM RCP8.5,
2086-2100



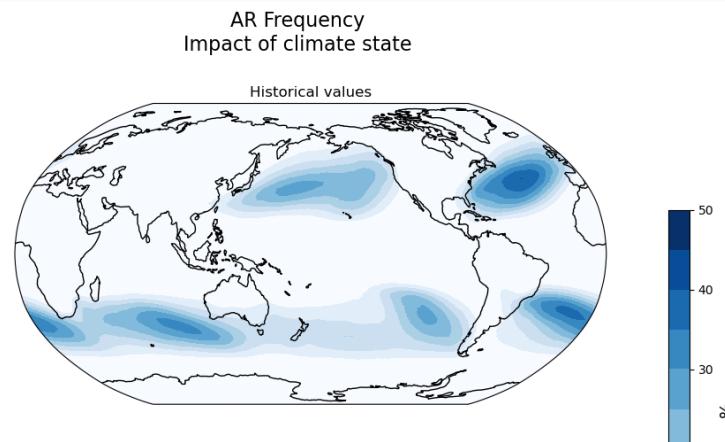
Comparing CESM results
across climate states.

What about the normalization
here?

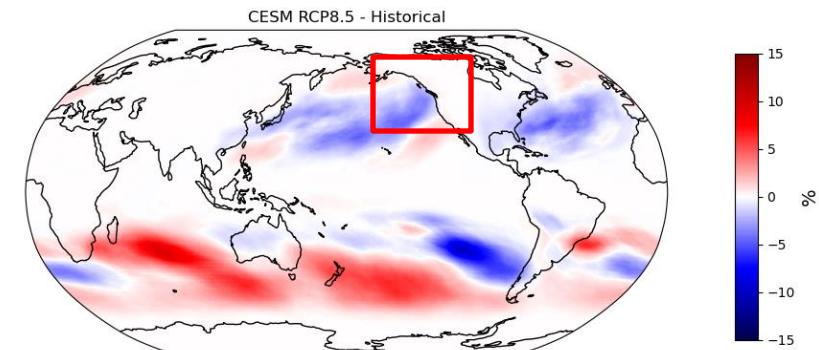
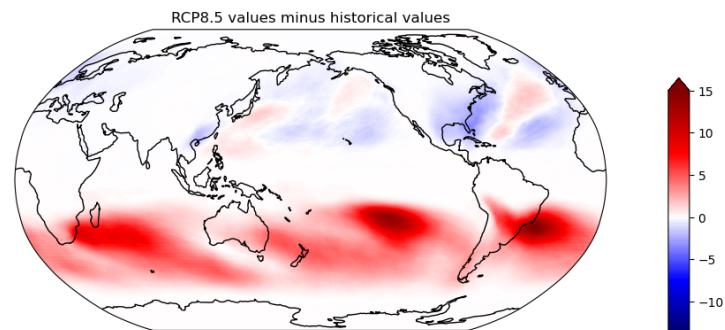
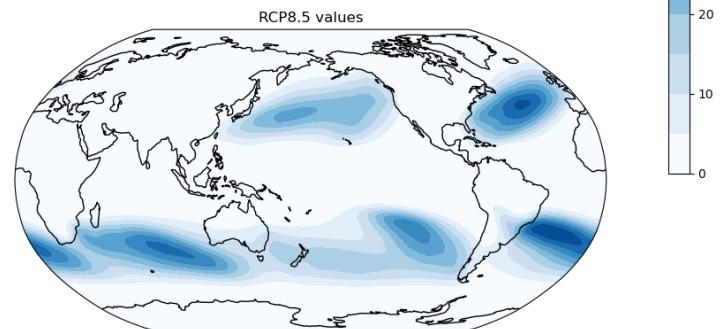
Dagon et al. *in prep*

Sensitivity to Normalization

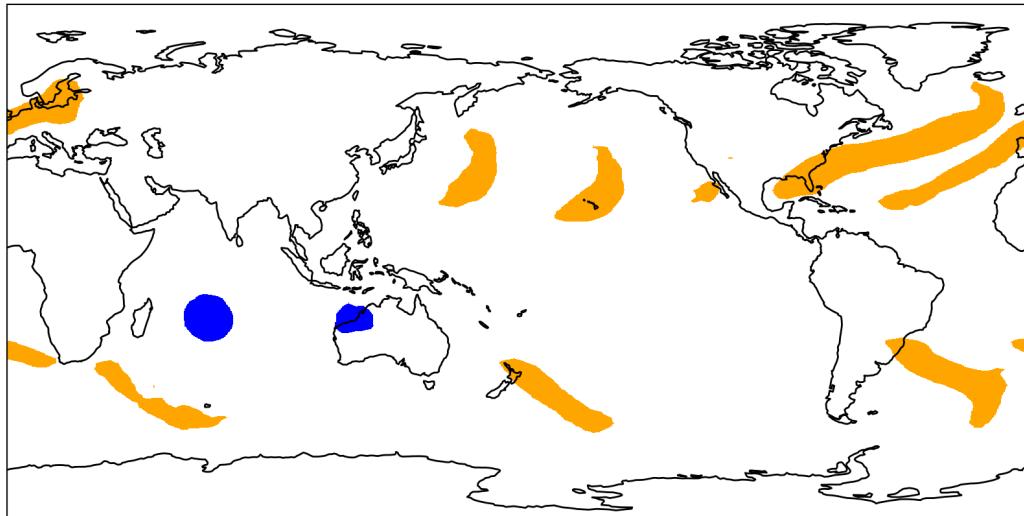
Calculate means/std from the training data (CAM)



Calculate means/std from the inference data (CESM RCP8.5)



Spatiotemporal Event Tracking



AR
TC
None

The screenshot shows a GitHub repository named 'ClimateNet' owned by 'andregraubner'. The 'Code' tab is selected, showing a file tree and the content of 'track_events.py'. The code is a Python script for tracking AR and TC events across time. It imports psutil, multiprocessing, tqdm, numpy, and xarray. The main function, 'track_events', takes class masks as xarray, minimum time length, tc drop threshold, ar drop threshold, and future lookup range as arguments. It omits components smaller than 250 pixels and events lasting less than 12 hours. The GitHub interface also shows issues, pull requests, actions, projects, security, and insights.

```
import psutil
from multiprocessing import Pool
from tqdm import tqdm
import numpy as np
import xarray as xr

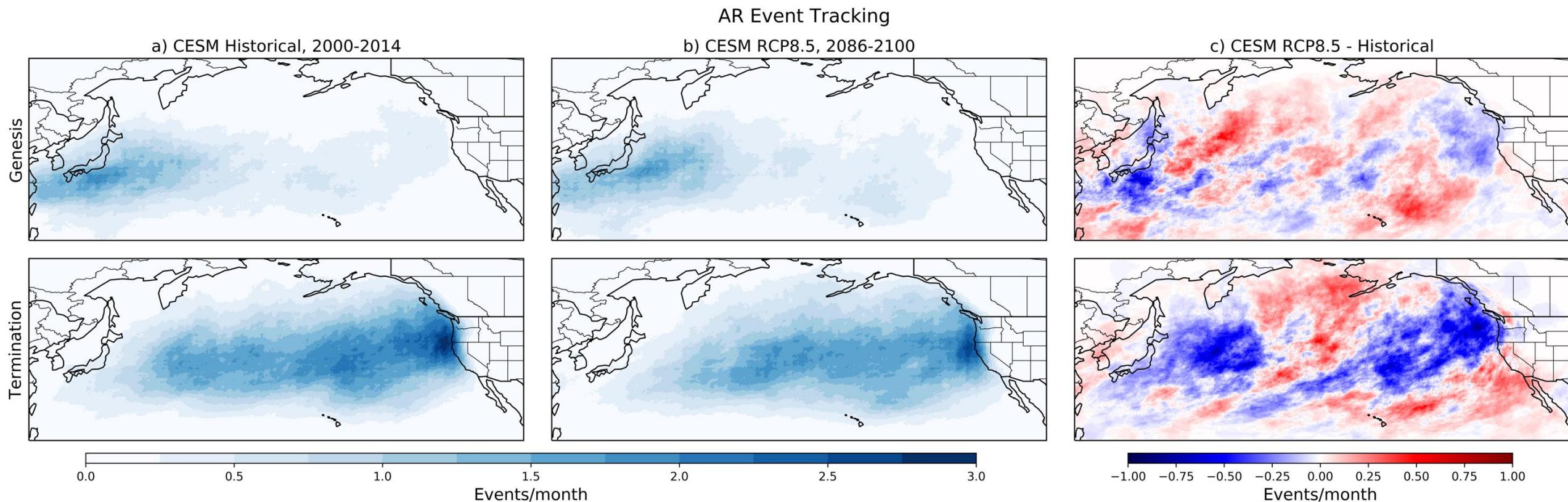
def track_events(class_masks_xarray, minimum_time_length=5,
                 tc_drop_threshold=250, ar_drop_threshold=250,
                 future_lookup_range=1):
    """track AR and TC events across time

    Keyword arguments:
    class_masks_xarray -- the class masks as xarray, 0==Background, 1==TC, 2 ==AR
    minimum_time_length -- the minimum number of time stamps an event ought to persist to be considered
    tc_dop_threshold -- the pixel threshold below which TCs are dropped
    ar_dop_threshold -- the pixel threshold below which ARs are dropped
    future_lookup_range -- across how many time stamps events get stitched together
    """

```

- Post-detection script iterates through samples in time and checks for connected components
- Assigns unique IDs to each event
- Omits components smaller than 250 pixels and events lasting less than 12 hours
- Very computationally intensive, so we adapted script to run for specific region (North Pacific basin)

Spatiotemporal Event Tracking



Peak genesis shifts northeastward in RCP8.5 relative to the historical simulation, while termination shifts southeastward along the California coast.

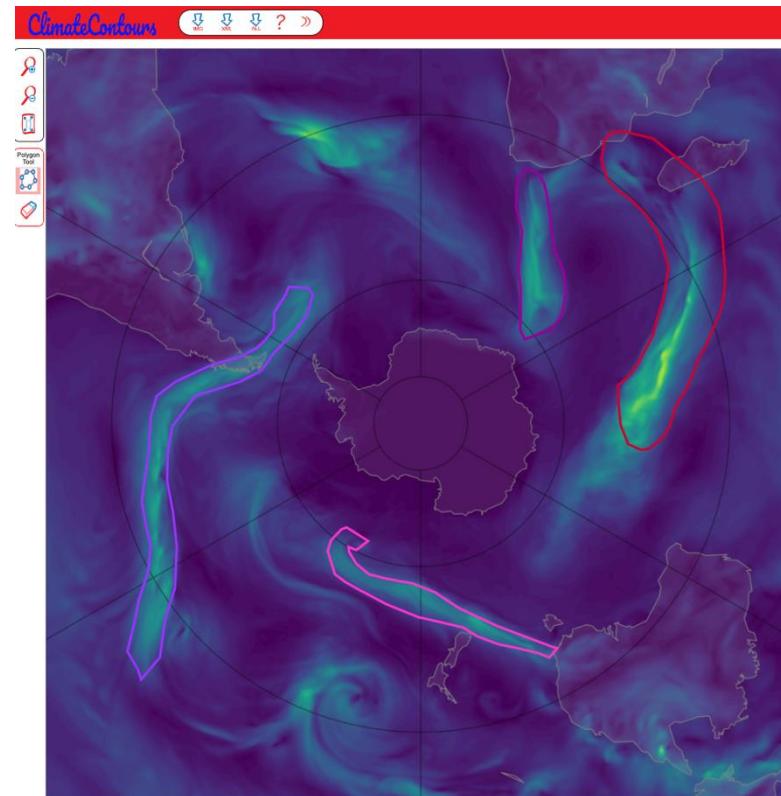
Dagon et al. *in prep*

Future Work: Machine Learning for Detection of Polar ARs

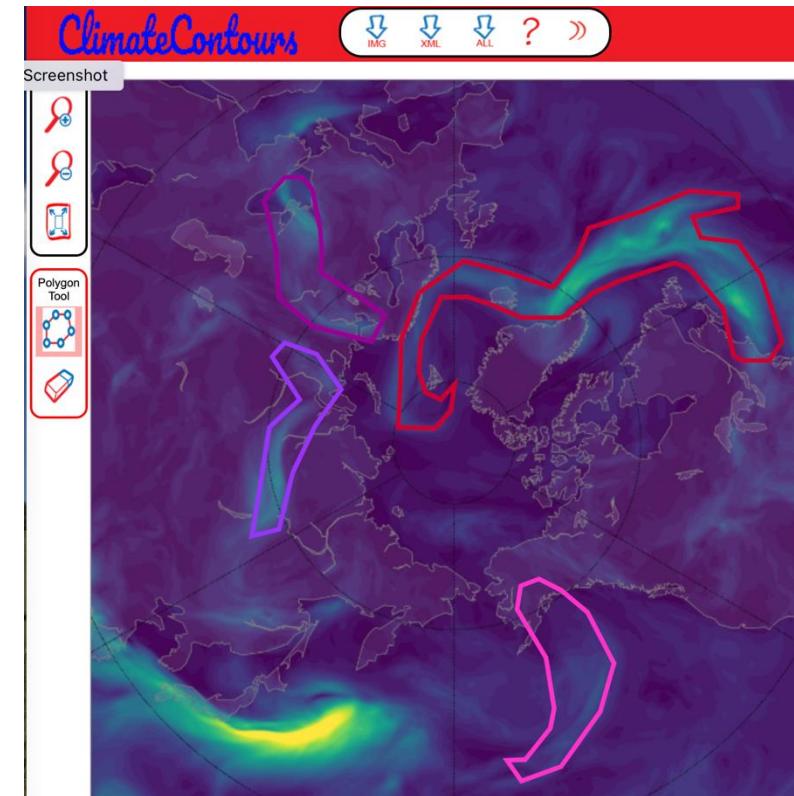


In development...polar AR algorithms for detecting Antarctic and Arctic atmospheric rivers

Antarctic Labeling Tool
~500 labels



Arctic Labeling Tool
~350 labels (so far)



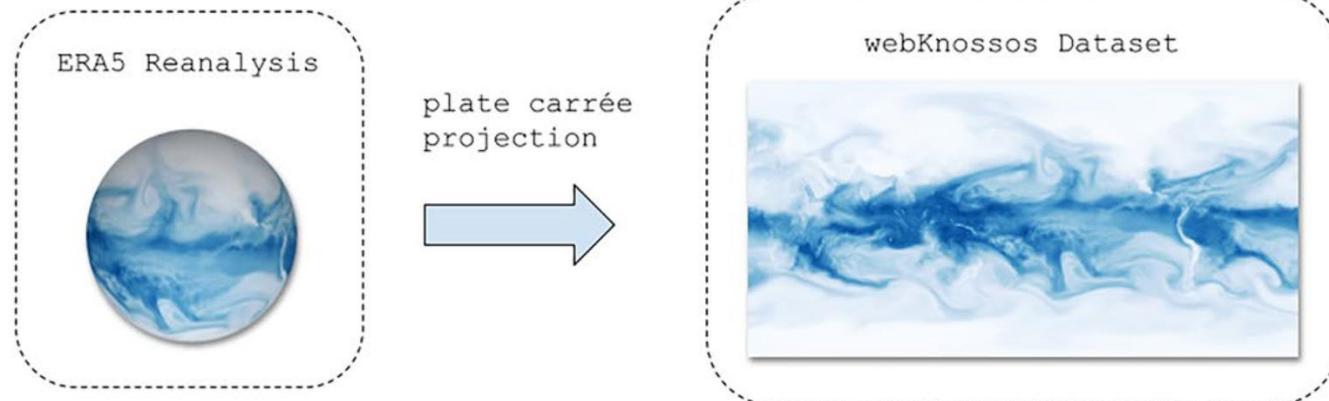
With: **Christine Shields, Teagan King, John Truesdale (NCAR CGD); Annette Greiner (LBNL), Sol Kim (Univ. Chicago)**

Future Work: Generating Large Amounts of Training Data



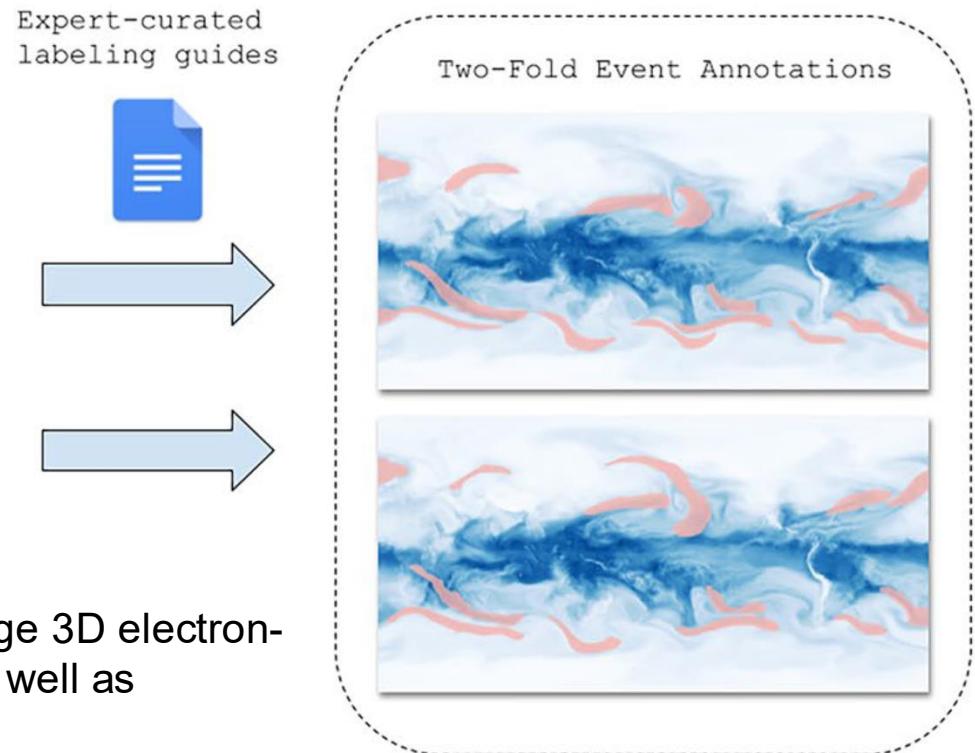
ClimateNetLarge

- Professional crowd-labeling (\$\$\$)
- ARs, TC, and blocking events
- Generated 49,184 hand-labeled timesteps!
- No ML models trained on this dataset (yet)



WEBKNOSSOS is an open-source tool for exploring large 3D electron-microscopy (EM) and light-microscopy (LM) datasets as well as **collaboratively generating annotations**.

Also includes a Python API for working with datasets and annotations:
<https://docs.webknossos.org/webknossos-py/>



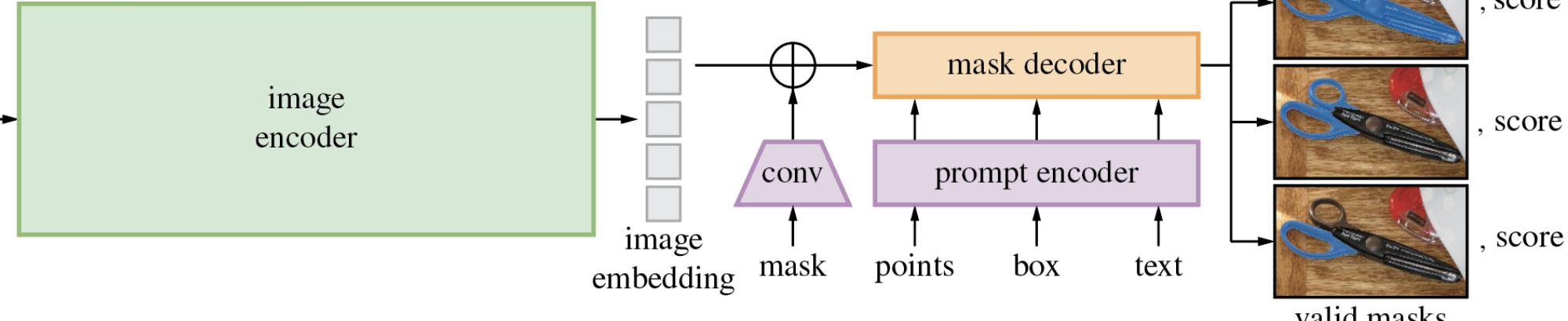
Kim et al. 2025

Future Work: Automated Segmentation Tools

Segment Anything: generate masks for all objects in an image



image



<https://github.com/facebookresearch/segment-anything>

Label Studio: open source data labeling platform



<https://labelstud.io/>

- ❖ **Machine learning image detection** applied to atmospheric rivers, tropical cyclones, and weather fronts in climate model simulations.
- ❖ Even with a pre-trained model, you still have to do **a lot of data processing!**
- ❖ **Leverage existing open-source code** and packages whenever possible.
- ❖ **Sensitivity tests** across inputs, resolutions, normalization, etc. are useful to understand the ML results.
- ❖ **Collaborations across Earth science and ML** made this work possible!

Thanks!  kdagon@ucar.edu
Questions?  [@katiedagon](https://github.com/katiedagon)