National Aeronautics and Space Administration

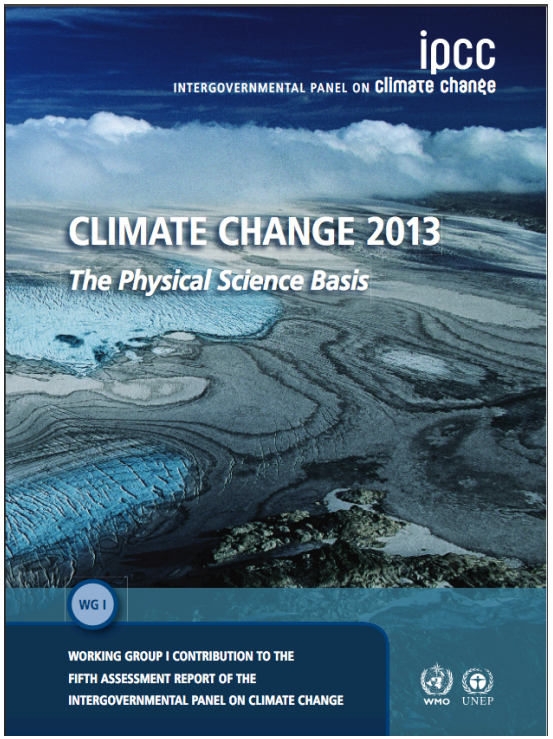
Machine Learning Analytic Services in EDAS

Thomas Maxwell, Thomas Favata, Dan Duffy, Laura Carriere, Jerry Potter

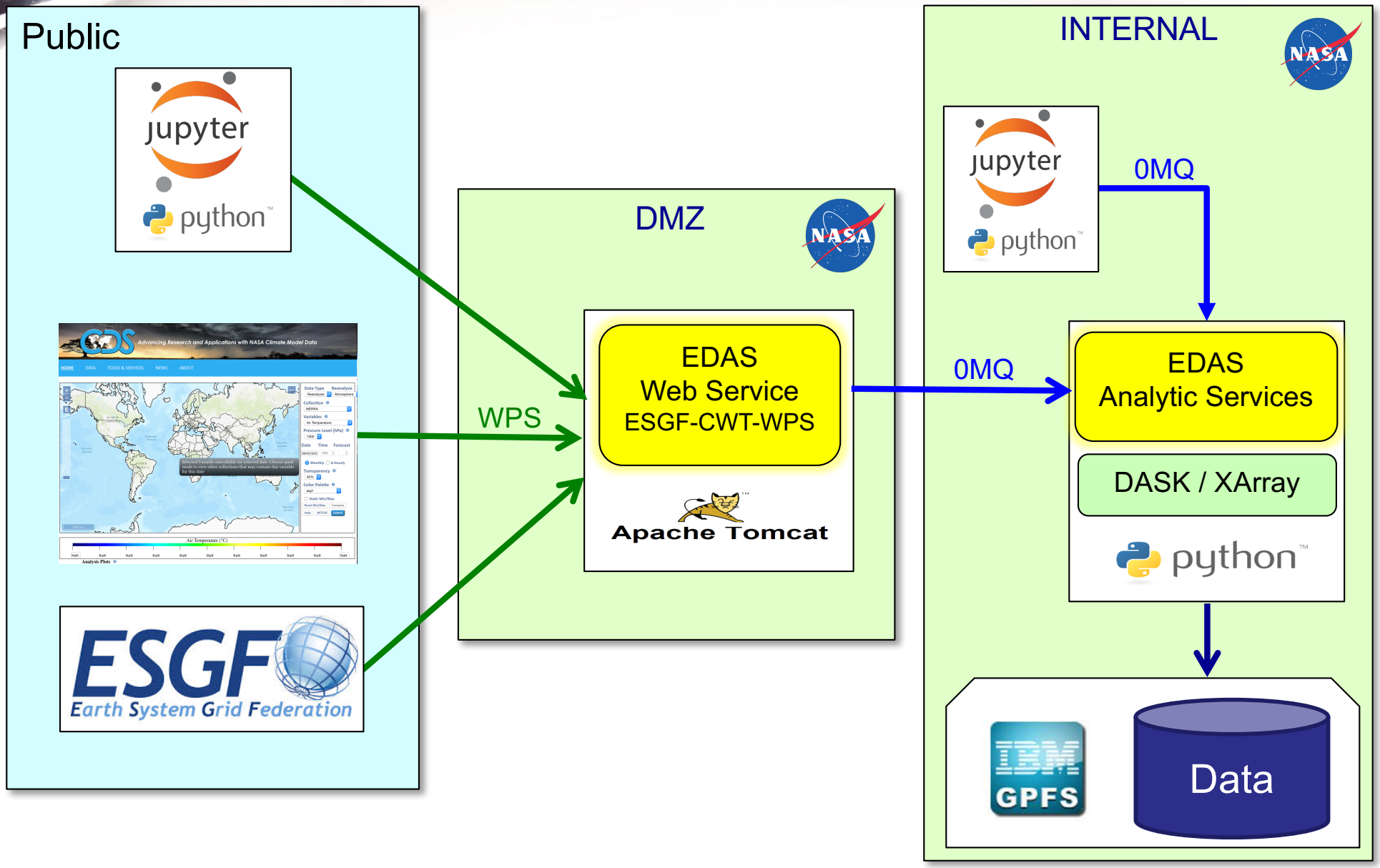
Earth System Grid Federation Compute Working Group

- ESGF distributes the data that supports the IPCC Assessment Reports
- CWT provides server-side analytics for ESGF
- CWT has defined a python API and a WPS service
- NASA-NCCS has implemented an ESGF-CWT analytics server (EDAS)
- Enables distributed, high-performance analytics close to the data

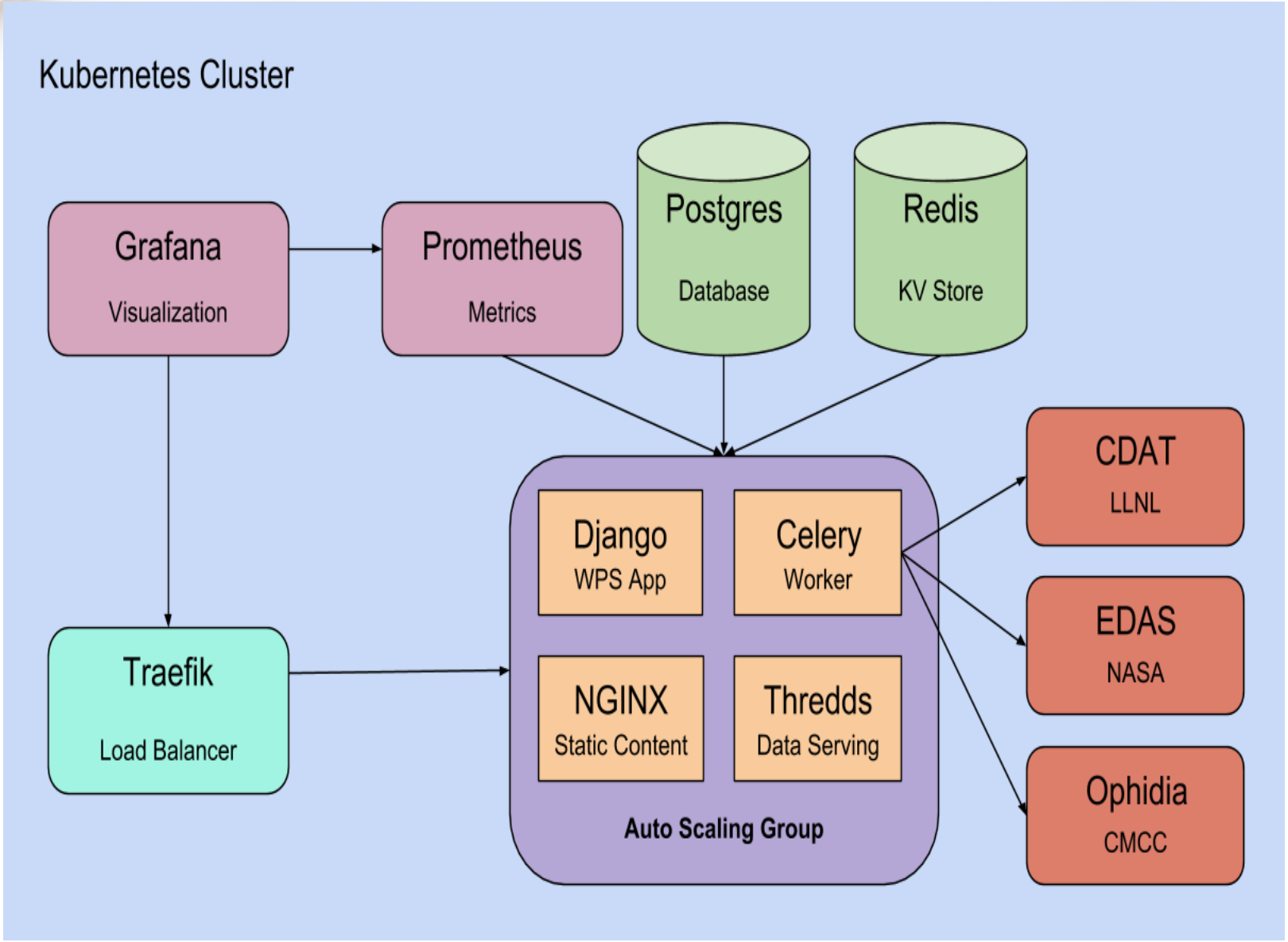
Estimated Data Growth of ESGF
2012 AR5 – 2 to 5 petabytes
2017 AR6 – 25 to 50 petabytes
2022 AR7 – 100 to 1,000 petabytes



NASA ESGF CWT Analytics (EDAS)



ESGF Analytics Architecture at LLNL



EDAS Analytic Services Framework

- Implemented in 100% python
  - Access to full python analytics ecosystem
- Built on Dask/Xarray
  - Dask-distributed parallelism
- Restful WPS interface
  - Esgf-cwt compliant
- Workflow framework
  - Compose graphs of canonical operations
- Parallel data access
  - Directly from POSIX or OpenDAP

Dask-distributed parallelism

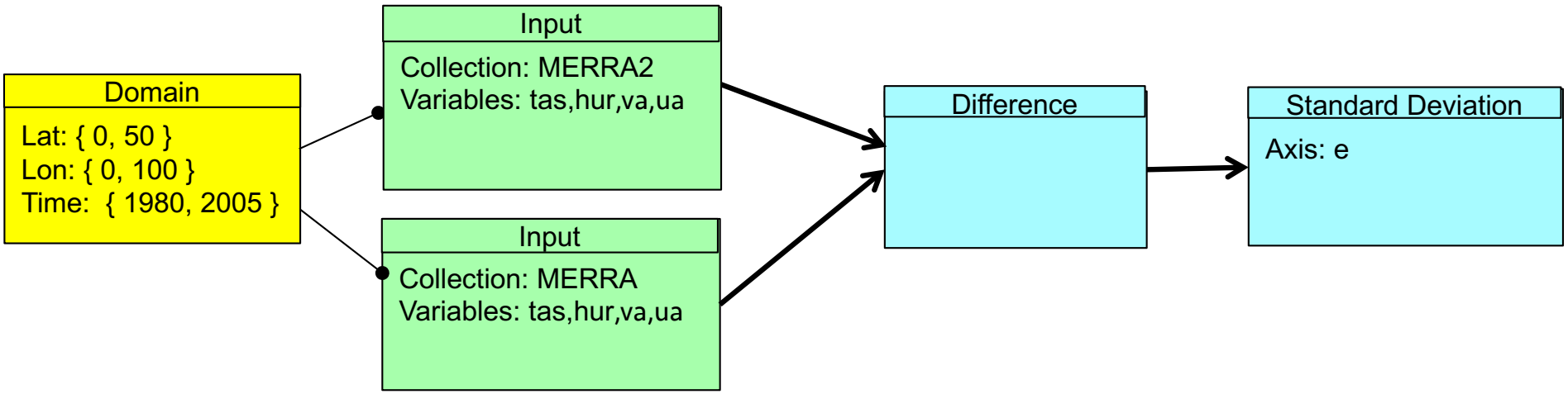
- Familiar APIs:
  - XArray builds on numpy and netCDF APIs.
  - High level constructs and automatic parallelism simplify development
- Pure Python:
  - Built in Python using well-known technologies
- Large group of developers
- Low latency:
  - Each task suffers about 1ms of overhead
- Peer-to-peer data sharing:
  - Workers communicate with each other to share data
- Complex Scheduling:
  - Supports complex workflows (not just map/filter/reduce)
- Data Locality:
  - Scheduling algorithms cleverly execute computations where data lives

Xarray Data Analysis Toolkit

- Extends Pandas to support N-dimensional arrays.
  - Inherits performance and power of Pandas.
- Tight integration with numpy and netCDF
  - In memory representation of netCDF data using np.ndarray
- Integrated with Dask for in-memory data parallel workflows
  - Transparent distributed (chunked) arrays
  - Lazy, streaming computation on datasets that don't fit in memory
  - Builtin parallel NetCDF IO
  - Automatically parallelizes xarray workflows
  - Parallelized numpy builtin and ufunc operations.

Request Structure

```
domains = [ {  name: d0,   lat:  {0, 50},   lon:  {0, 100},  time: { '1980-01-01','2005-01-01' } } ]
variables = [          { col: merra,    name: tas,hur,va,ua,   domain: d0, result: v0  }
                       { col: merra2,   name: tas,hur,va,ua,   domain: d0, result: v1  } ]
operations = [         { name: xarray.diff, input: v0,v1, result: vdiff   }
                       { name: xarray.std, input: vdiff, axes: e }    ]
```



Kernels

Canonical operations:

- Data access & subset
- Average (weighted and unweighted)
- Maximum
- Minimum
- Sum
- Difference
- Product
- Standard Deviation
- Variance
- Anomaly
- Median
- Norm
- Filter
- Decycle
- Highpass/Detrend
- Lowpass/Smooth

Specialized operations:

- EOF
- PC
- Climate Indices
- Teleconnection Map
- Neural Network Kernels:
  - Layer
  - Trainer
  - Model

Canonical Operation Options

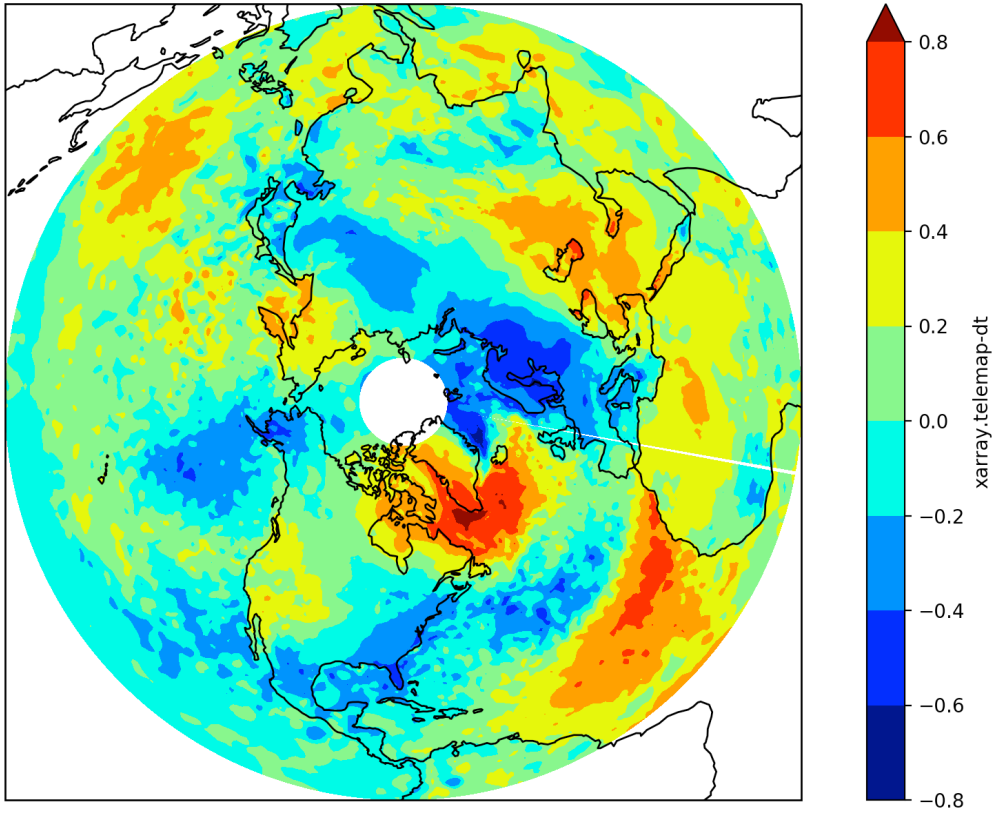
- Domain: subset to region of interest
- Axes: reduce over axes
  - X (latitude), Y (longitude) , Z (levels), T (time), E (ensemble)
- Groupby: split-apply-combine
  - Custom or existing Axis
  - Pandas groups
- Resample: upsampling and downsampling
  - Pandas resample API

Example (for 10 years of data):

Operation	Interpretation	Size
ave(axis: t)	Time average	1
ave(axis: te)	Time ensemble average	1
ave(axis: t, groupby: t.month)	Monthly climatology	12
ave(axis: t, resample: t.month)	Monthly means	120

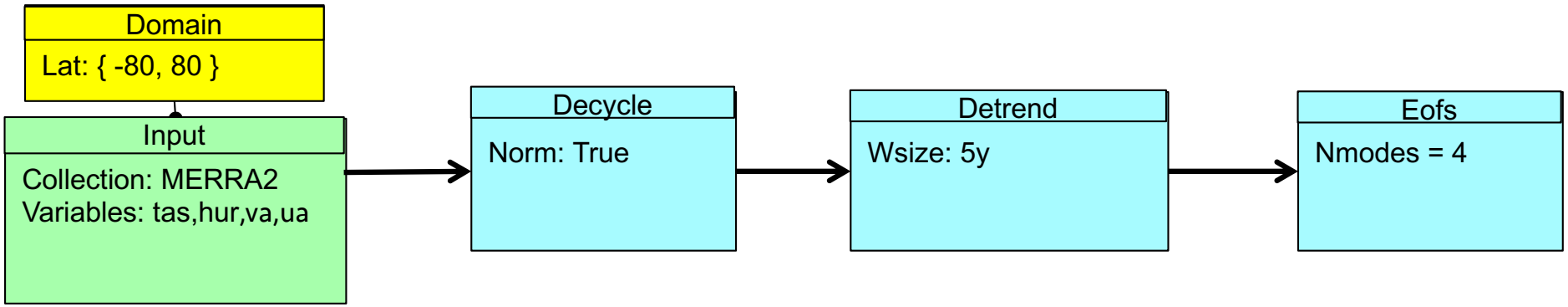
Teleconnection Maps

Computes a map of covariances between a chosen point and all other points in the ROI.

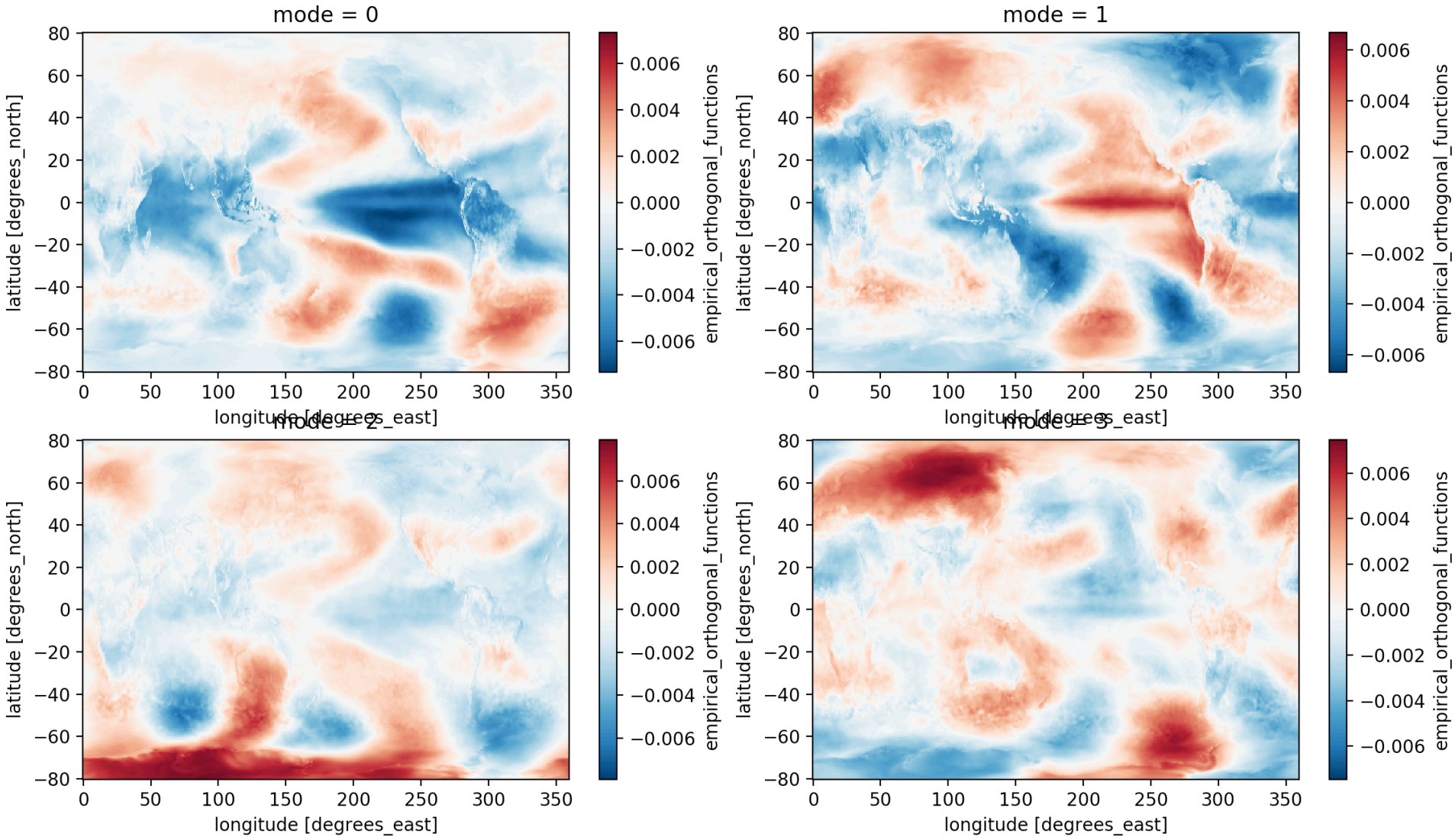


EOF Workflow

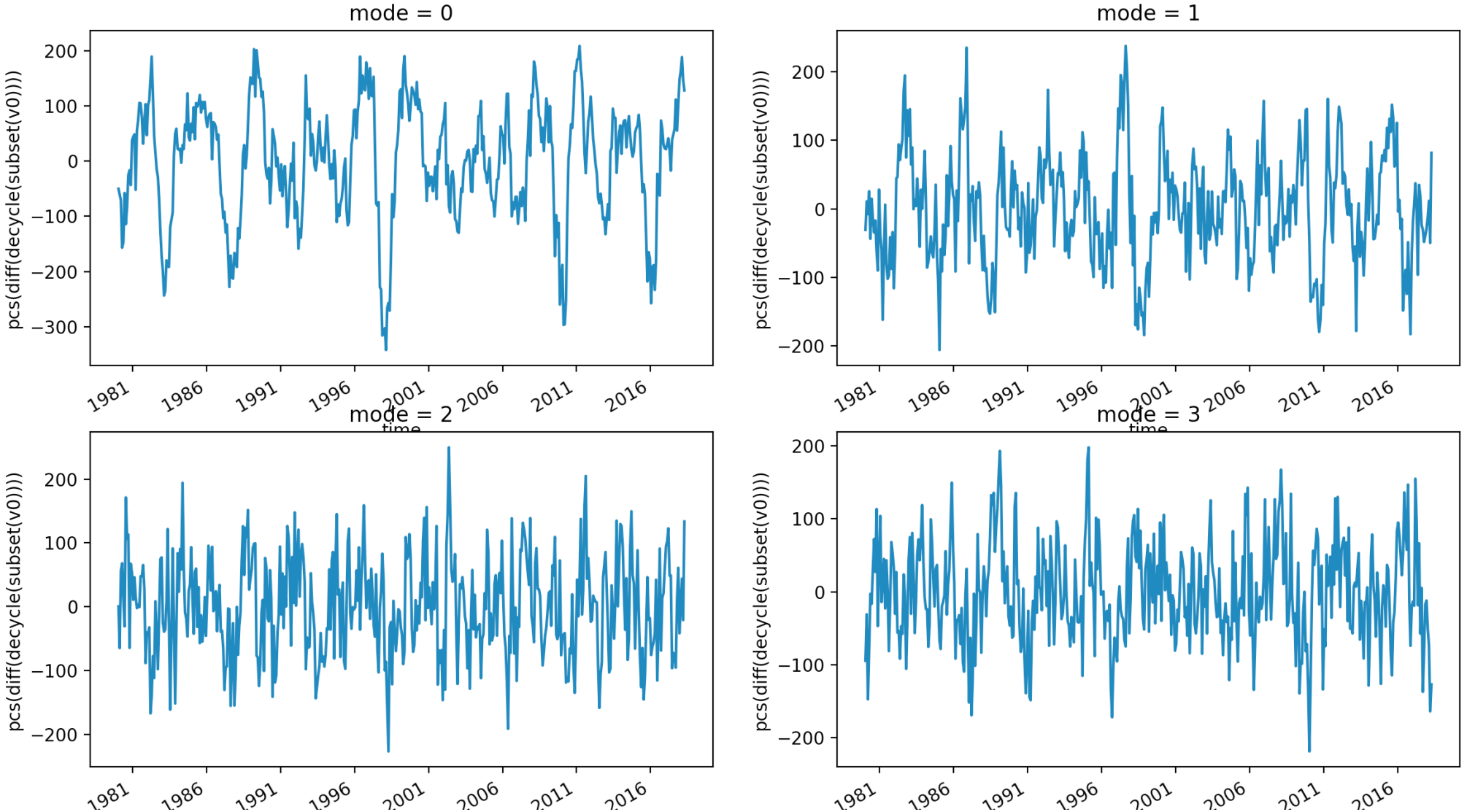
EOF Decomposition: $$X(t, \mathbf{s}) = \sum_{k=1}^{M} c_k(t) u_k(\mathbf{s}),$$



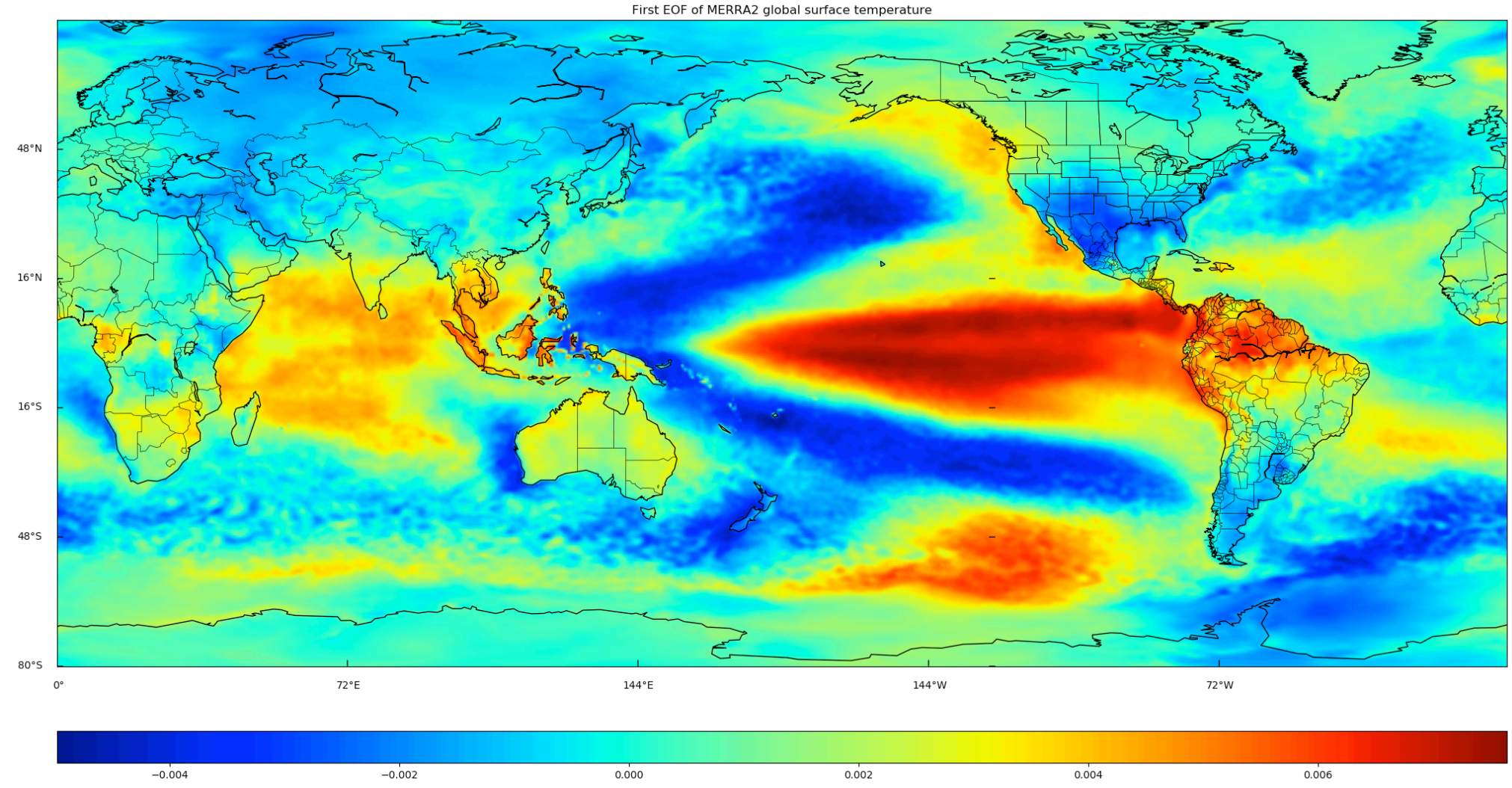
MERRA2 Global Surface Temperature
EOF Modes



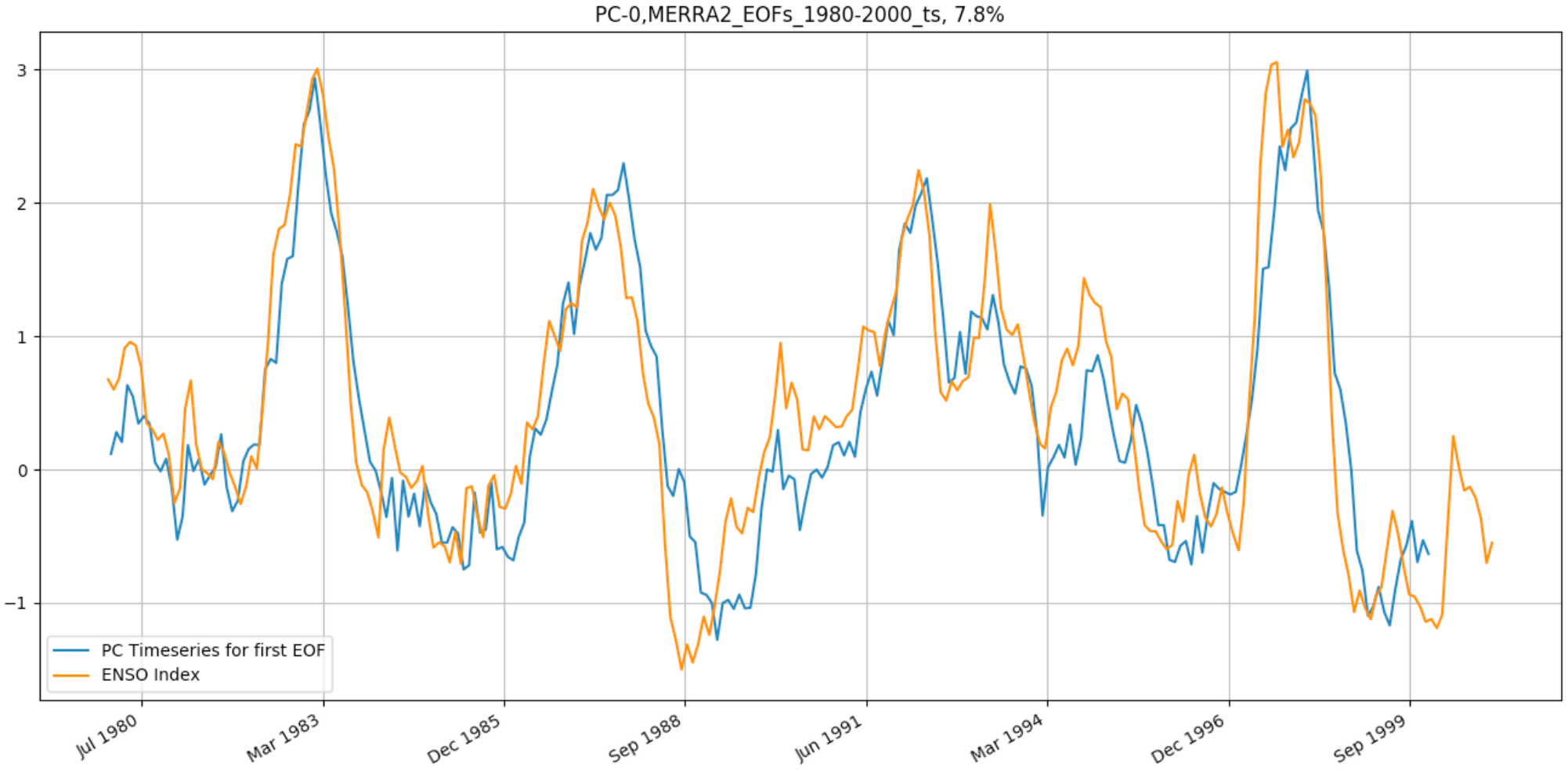
MERRA2 Global Surface Temperature
Principal Component Timeseries



MERRA2 Global Surface Temperature First EOF

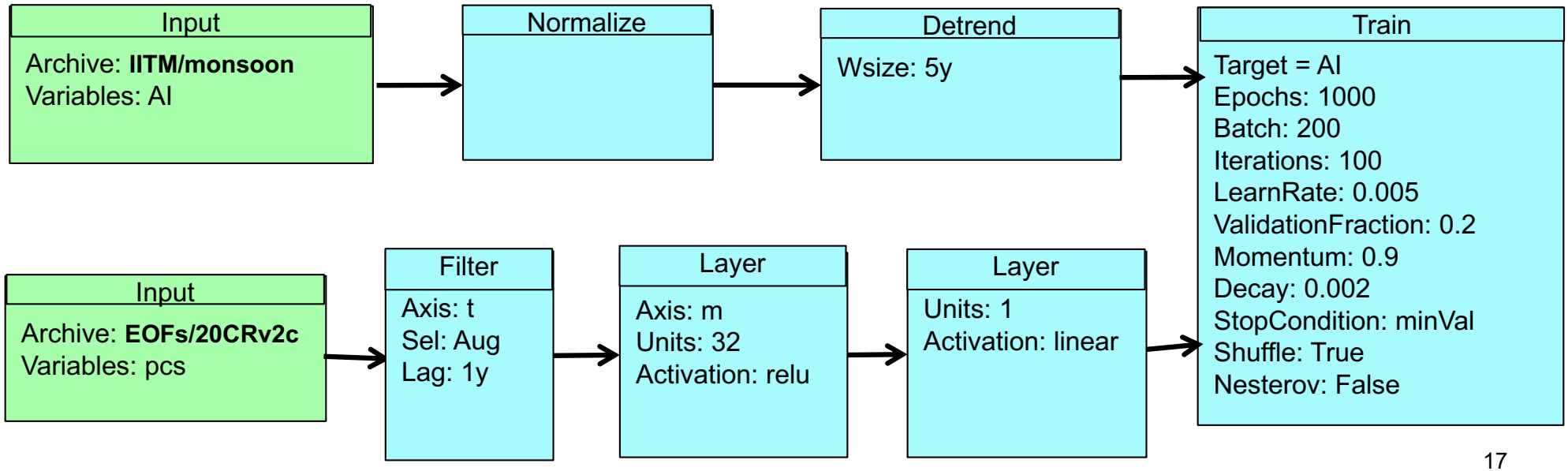


PC1 – ENSO Index Comparison



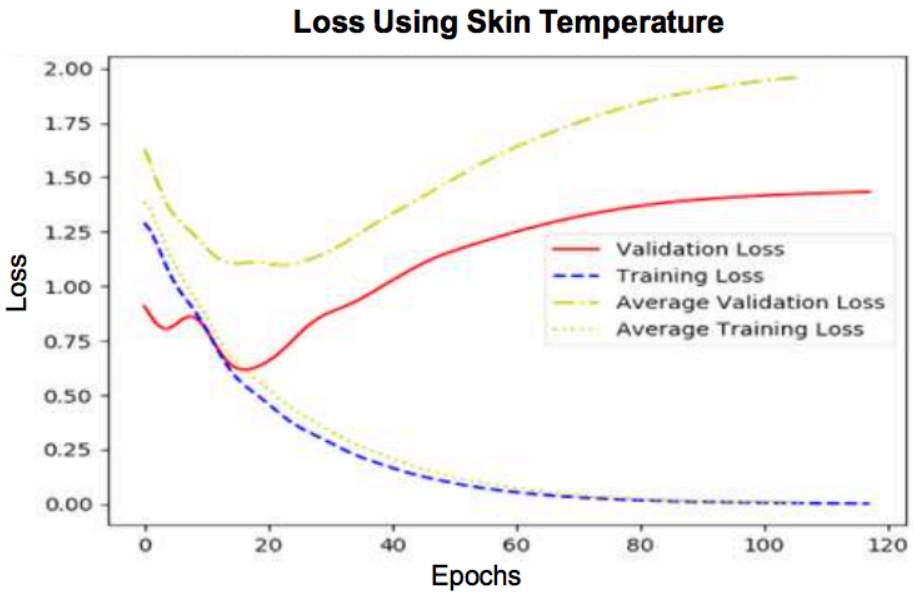
Machine Learning Workflow

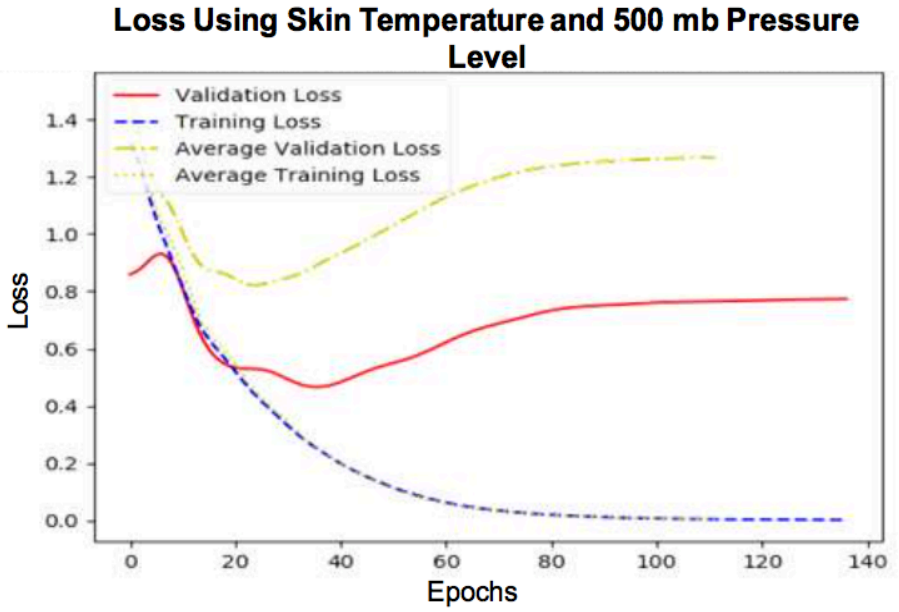
- Predict All-India Monsoon rainfall accumulation one year in advance
- Use a two-layer neural network
- Inputs: First 16 PCs of surface temperature & 500 mbar height
  - 1 year lag time (August values)



Training Performance

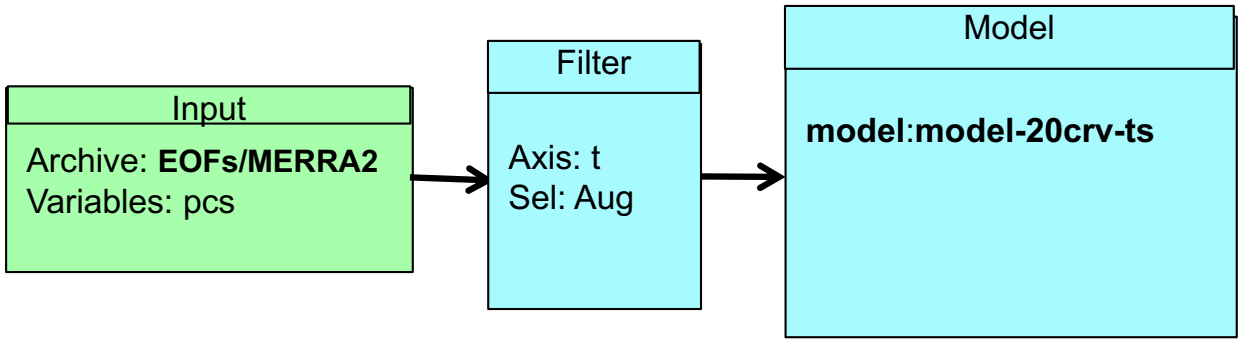
- Loss Function: Mean square error
  - Output node results vs. IITM-AI timeseries
- Last 20% of data reserved for validation
- Choose model with minimum error on validation data





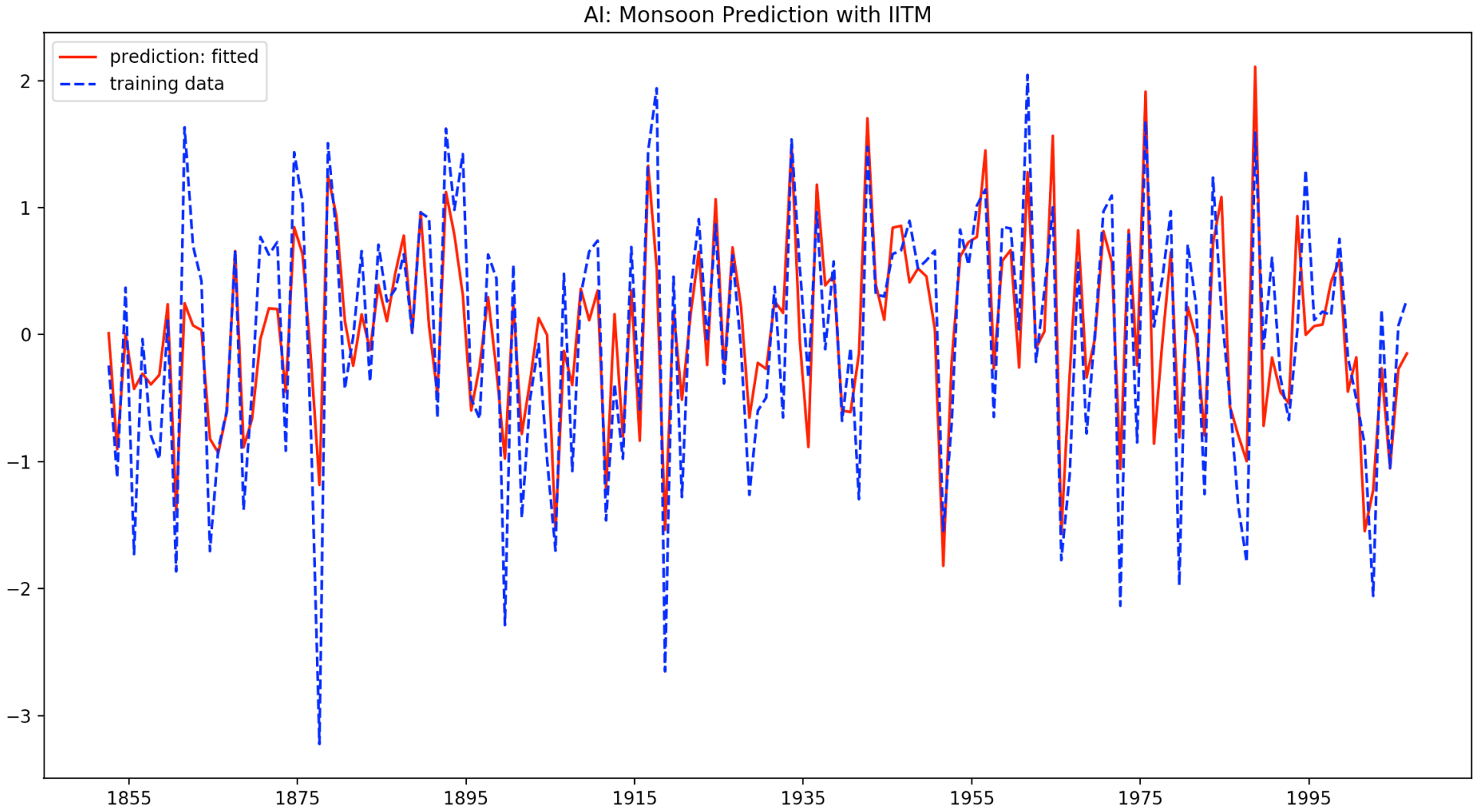
Applying the Neural Network Model

- Model kernel reads generated network structure and weights
- Generates a projection from a set of PCs



Results

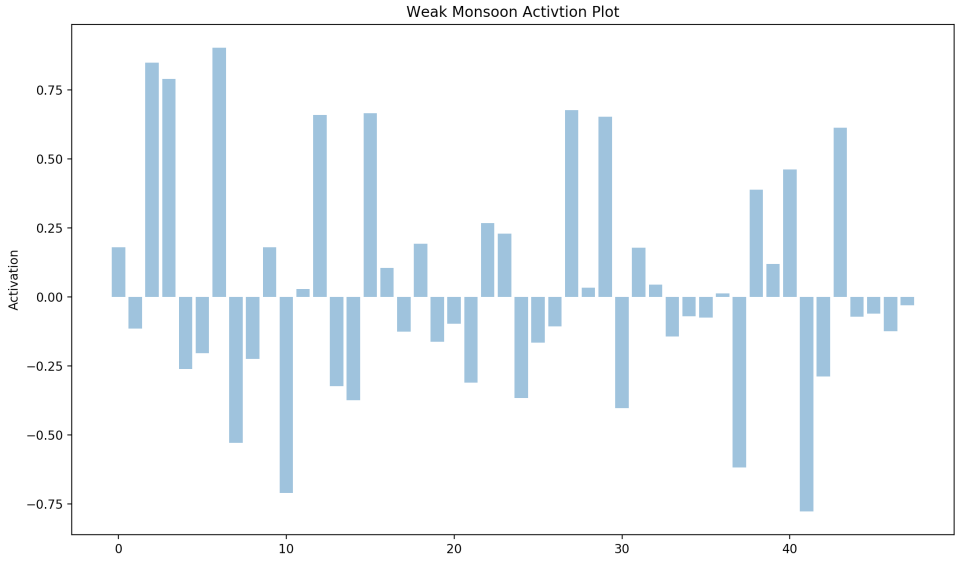
- Comparison of predicted to actual monsoon precipitation
- Result of two month project by summer intern



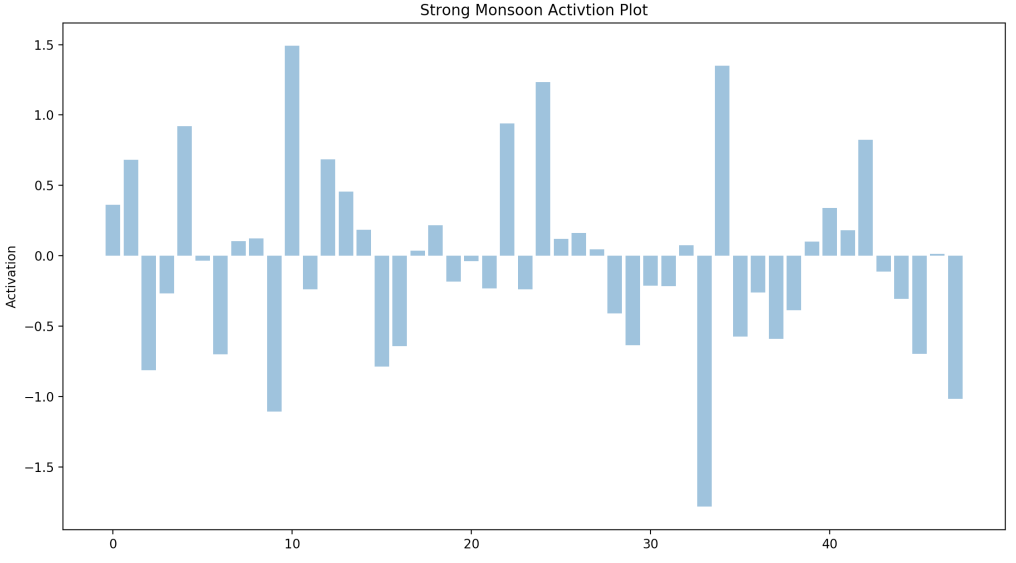
Backproject Activation Patterns

Use network gradient to compute optimal input patterns for given outputs.

Weak Monsoon Activation Pattern

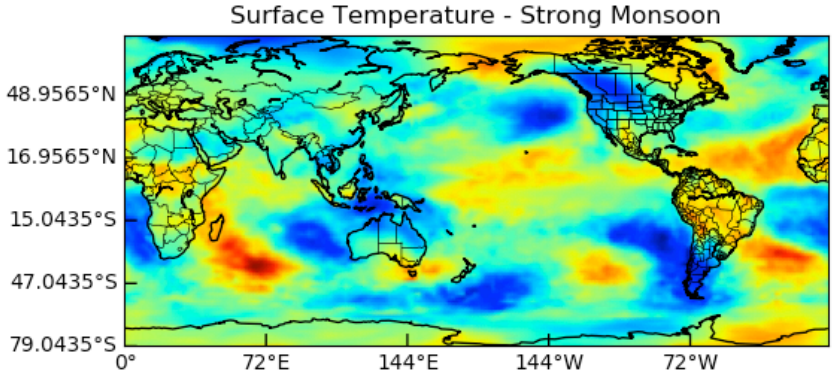
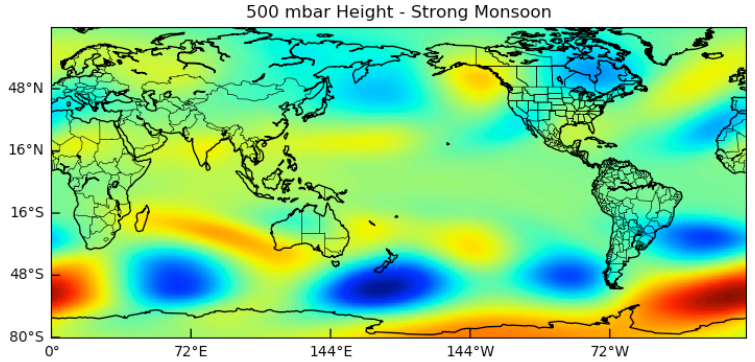
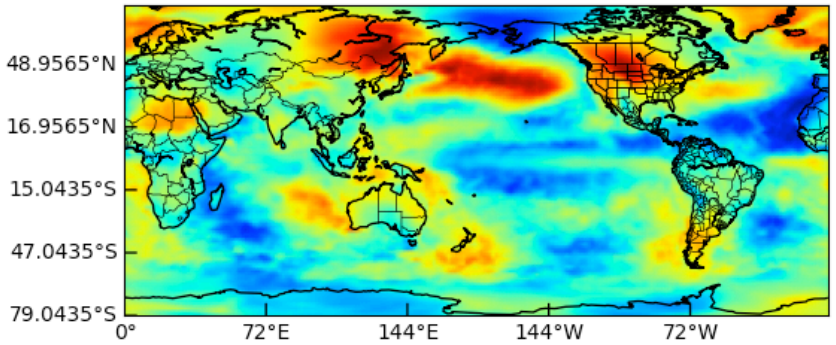
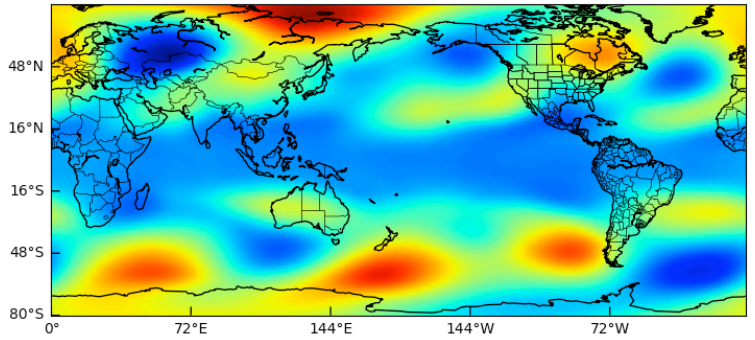
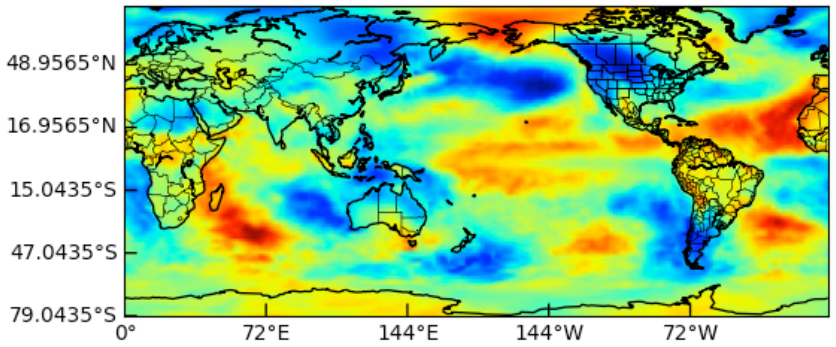
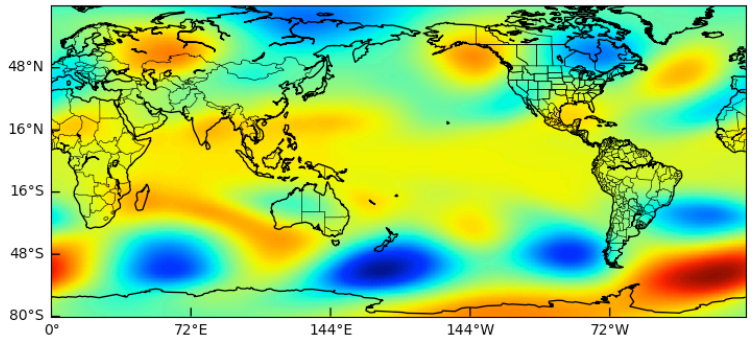


Strong Monsoon Activation Pattern



Backproject Input Patterns

Compute optimal input patterns for given outputs.

	Surface Temperature	500 mbar Height
Strong Monsoon		
Weak Monsoon		
Difference		

Conclusions

- Server side analytics will increasing play a crucial role in the earth data scientist's toolkit.
- Workflow frameworks enable flexible and powerful access to high performance computing resources.

Inquires: thomas.maxwell@nasa.gov

EDAS: https://github.com/nasa-nccs-cds/edask.git