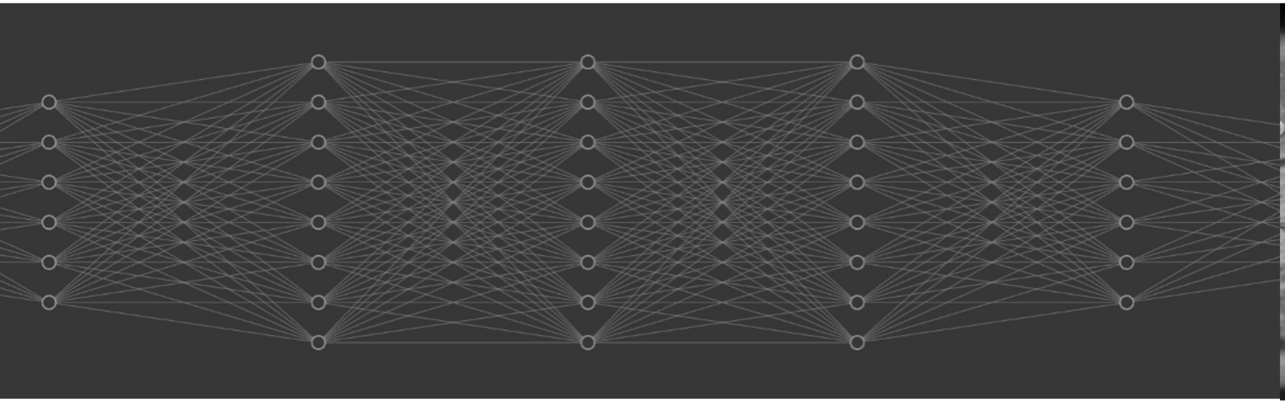
Computational Needs for Deep Learning Prediction of Global Precipitation



Maria J. Molina

NCAR CGD, Boulder, Colorado

NCAR U.S. DEPARTMENT OF ENERGY | Office of Science NSF

catalyst

A. Project Information

Title of Project:

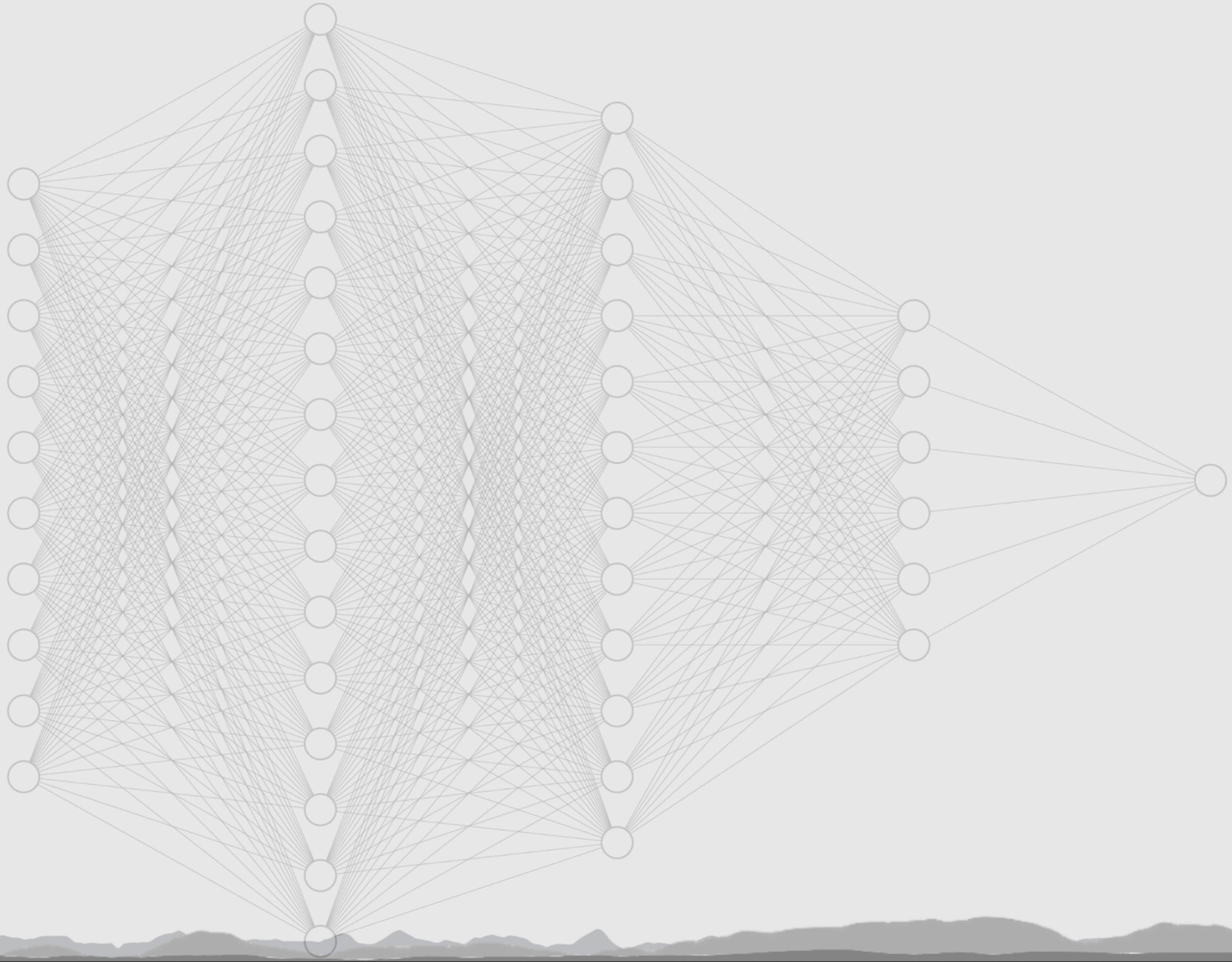
Deep Learning-based Large Ensemble for Subseasonal Prediction of Global Precipitation

Project Lead: Maria J. Molina, Project Scientist I, NCAR CGD, Boulder, CO

Project Co-Lead: Katherine (Katie) Dagon, Project Scientist I, NCAR CGD, Boulder, CO

Submission Date: January 24, 2022

Collaborators: Jadwiga Richter (CGD), Judith Berner (MMM), David John Gagne (CISL), Gerald Meehl (CGD), John Schreck (CISL), William Chapman (ASP), Aixue Hu (CGD), Anne Glanville (CGD), and Abby Jaye (MMM).



1959, ML defined

1986,

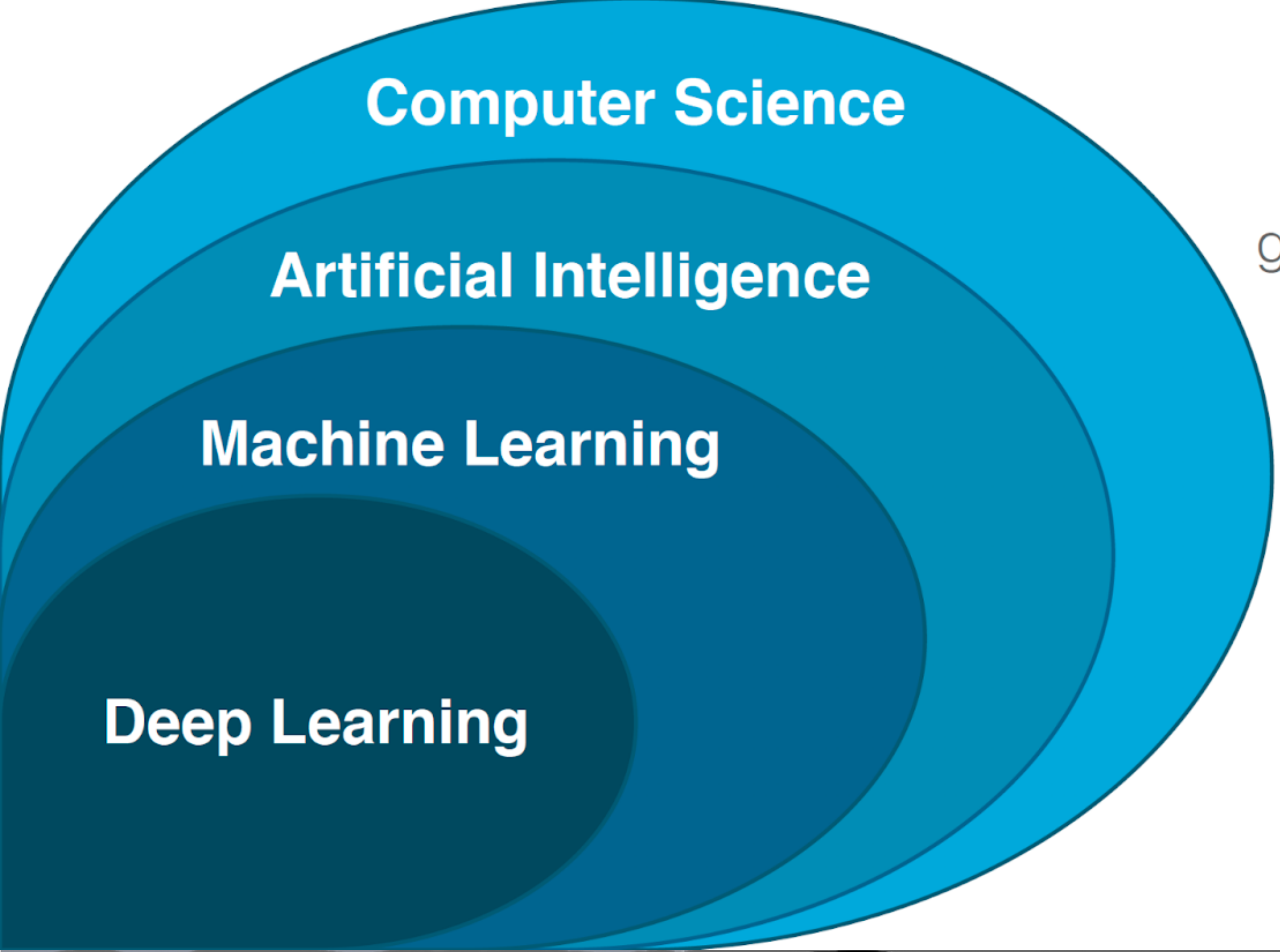
Backpropagation

Since 1990s,

GPUs

ImageNet

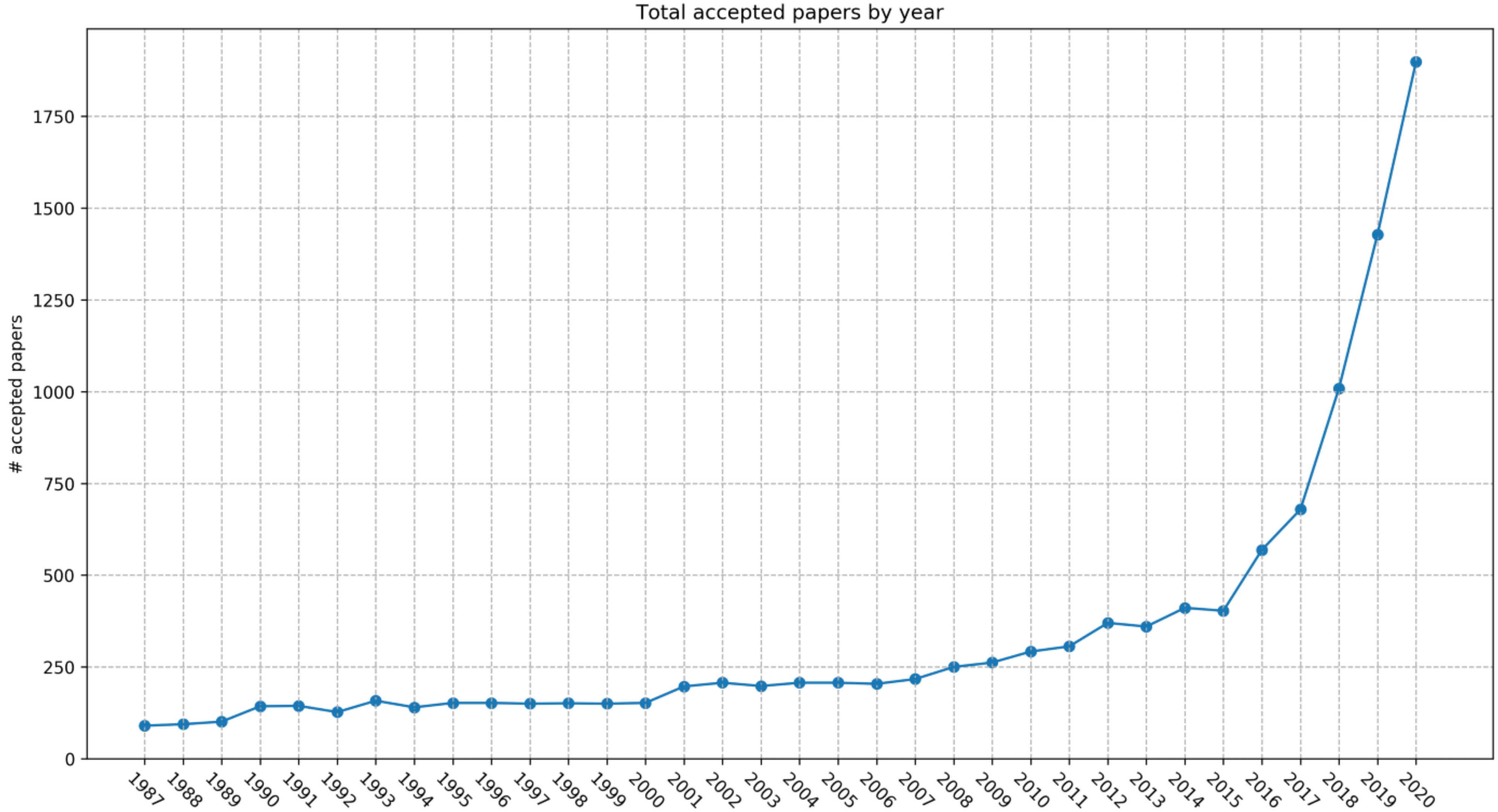
DL advances



"Field of study that gives computers the ability to learn without being explicitly programmed."

1959, Arthur Lee Samuel defined ML

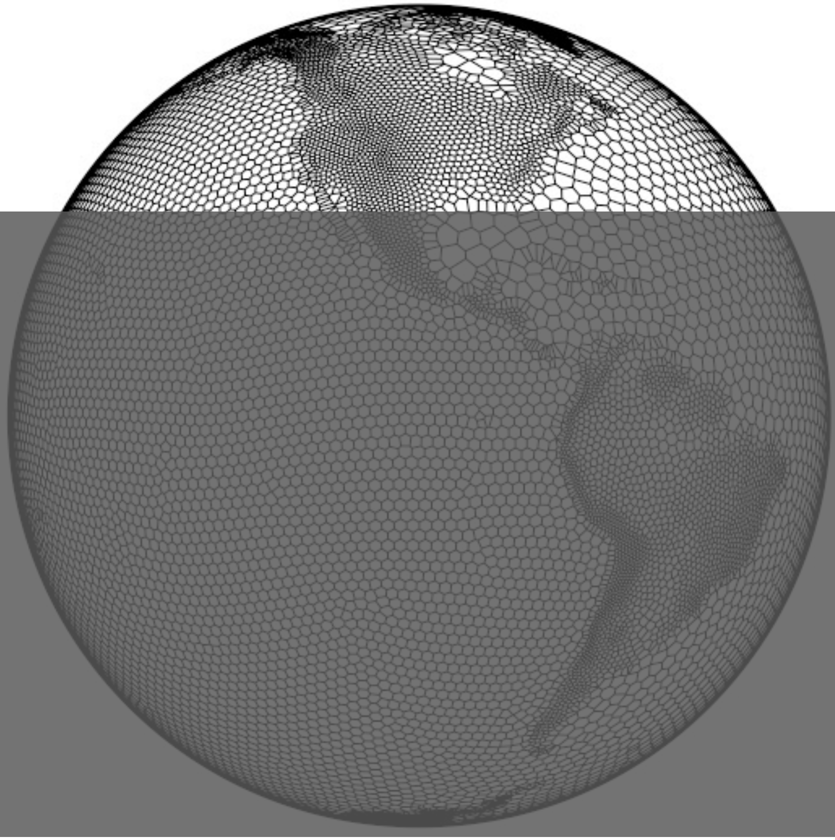
NeurIPS Conference Papers, 1987-2020



NCAR U.S. DEPARTMENT OF ENERGY Office of Science NSF catalyst

ML for Earth system modeling should incorporate:

- Physics and Domain Knowledge
- Robustness
- Interpretable ML and Explainable AI

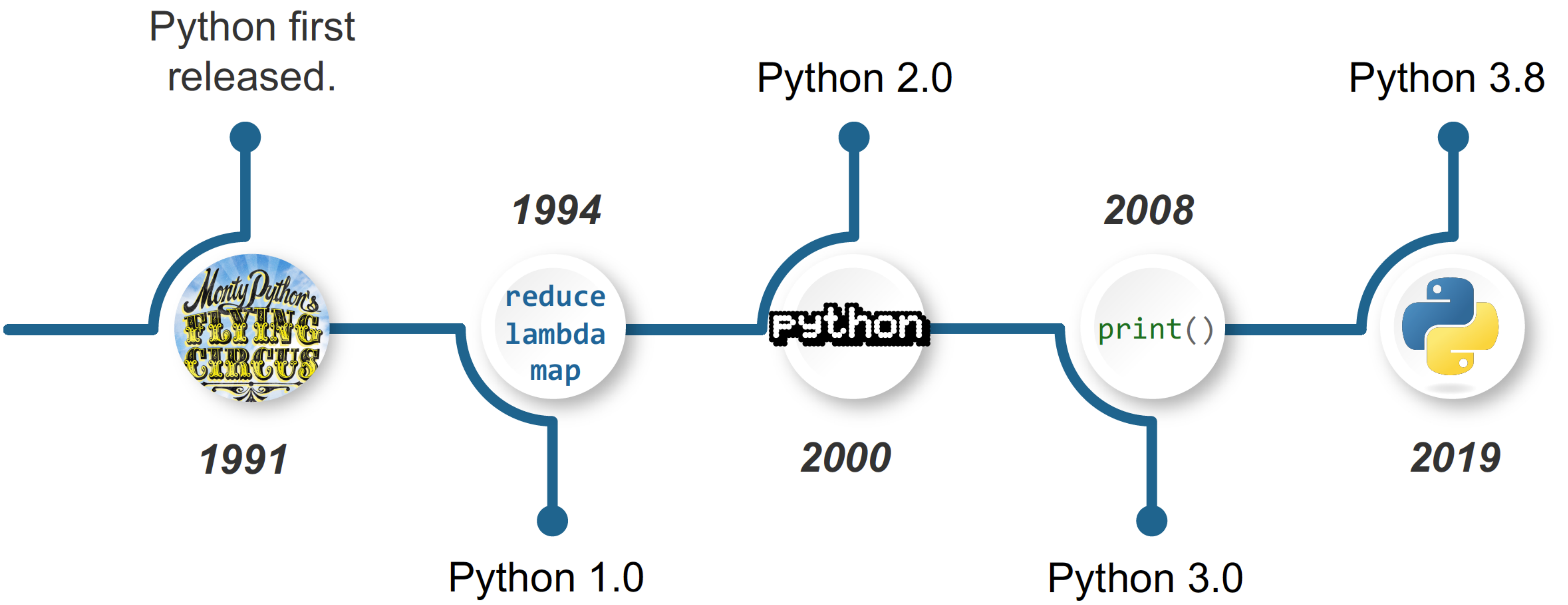


python

PyTorch

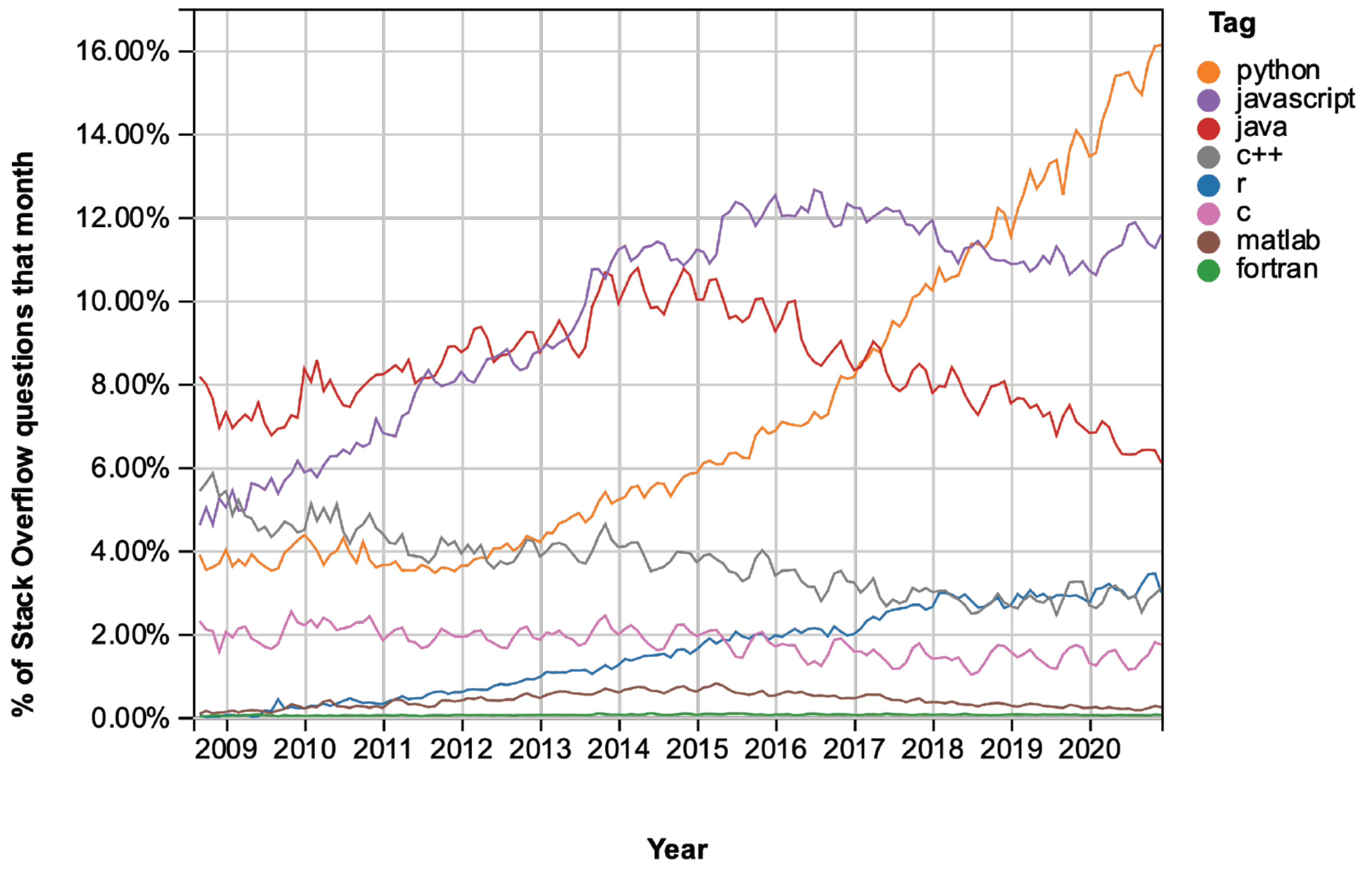
Keras

TensorFlow


Python first released.
1991
1994
reduce
lambda
map
Python 1.0
Python 2.0
python
2000
2008
print()
Python 3.0
Python 3.8
2019

python.org/

stackoverflow trends



https://insights.stackoverflow.com/trends?tags=r%2Cpython%2Cfortran%2Cjava%2Cjavascript%2Cmatlab%2Cc%2Cc%2B%2B xarray

pandas

matplotlib

bokeh

MetPy

NumPy

DASK

SciPy

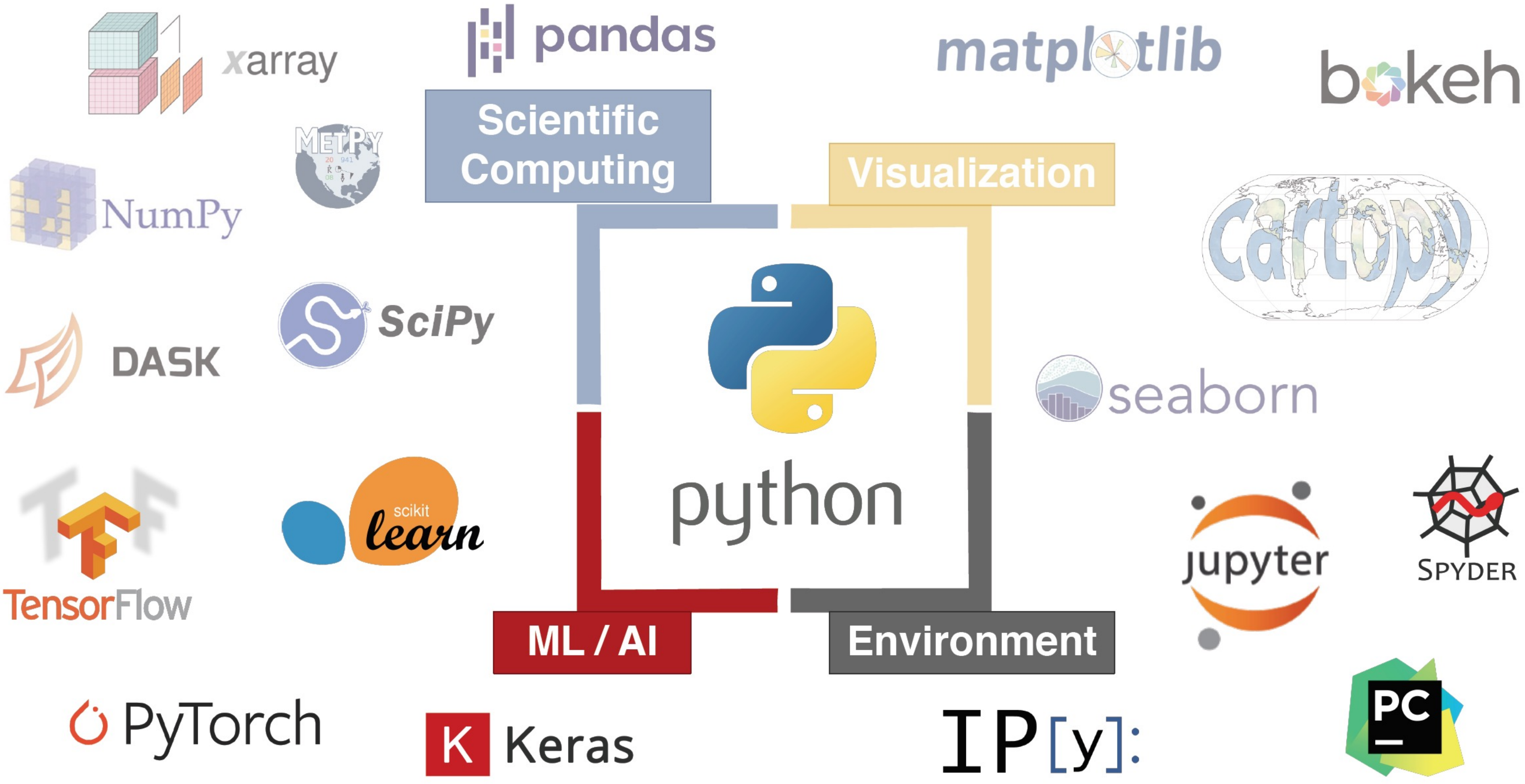
Scientific Computing

Visualization

python

Cartopy

seaborn


xarray
pandas
matplotlib
bokeh
Scientific Computing
Visualization
MetPy
NumPy
cartopy
SciPy
DASK
python
seaborn
TensorFlow
scikit learn
jupyter
Spyder
ML / AI
Environment
PyTorch
Keras
IP[y]:
PC

A. Project Information

Title of Project:

Deep Learning-based Large Ensemble for Subseasonal Prediction of Global Precipitation

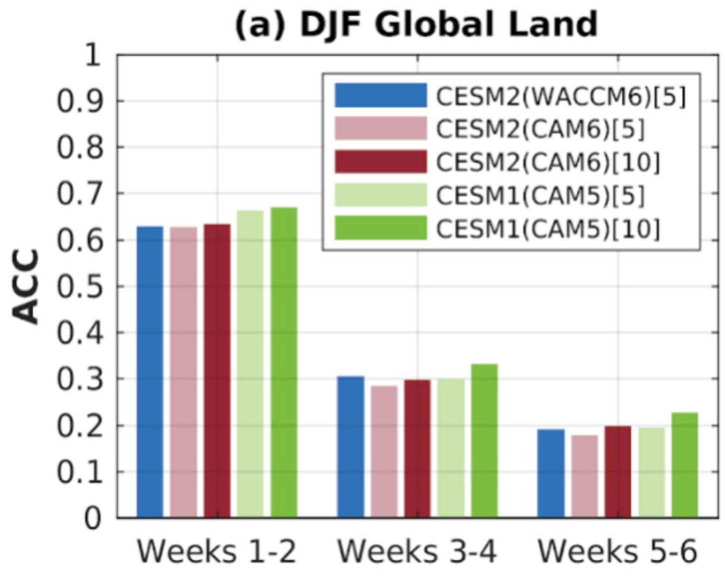
Project Lead: Maria J. Molina, Project Scientist I, NCAR CGD, Boulder, CO

Project Co-Lead: Katherine (Katie) Dagon, Project Scientist I, NCAR CGD, Boulder, CO

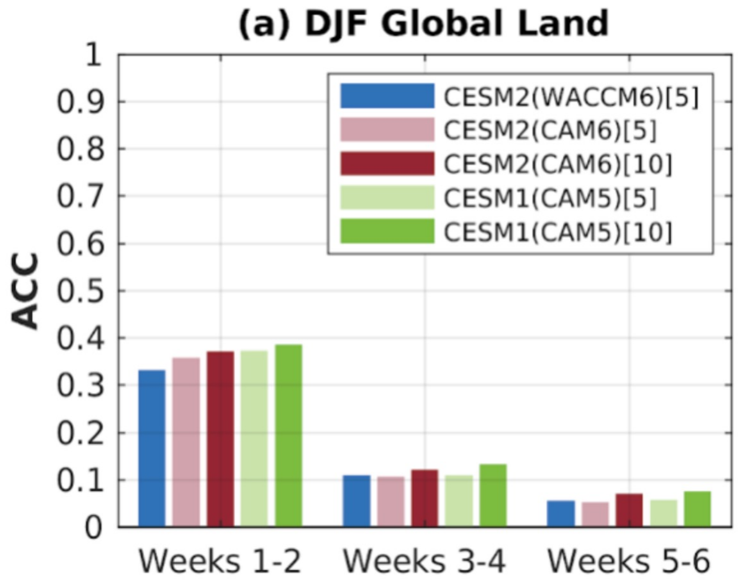
Submission Date: January 24, 2022

Collaborators: Jadwiga Richter (CGD), Judith Berner (MMM), David John Gagne (CISL), Gerald Meehl (CGD), John Schreck (CISL), William Chapman (ASP), Aixue Hu (CGD), Anne Glanville (CGD), and Abby Jaye (MMM).

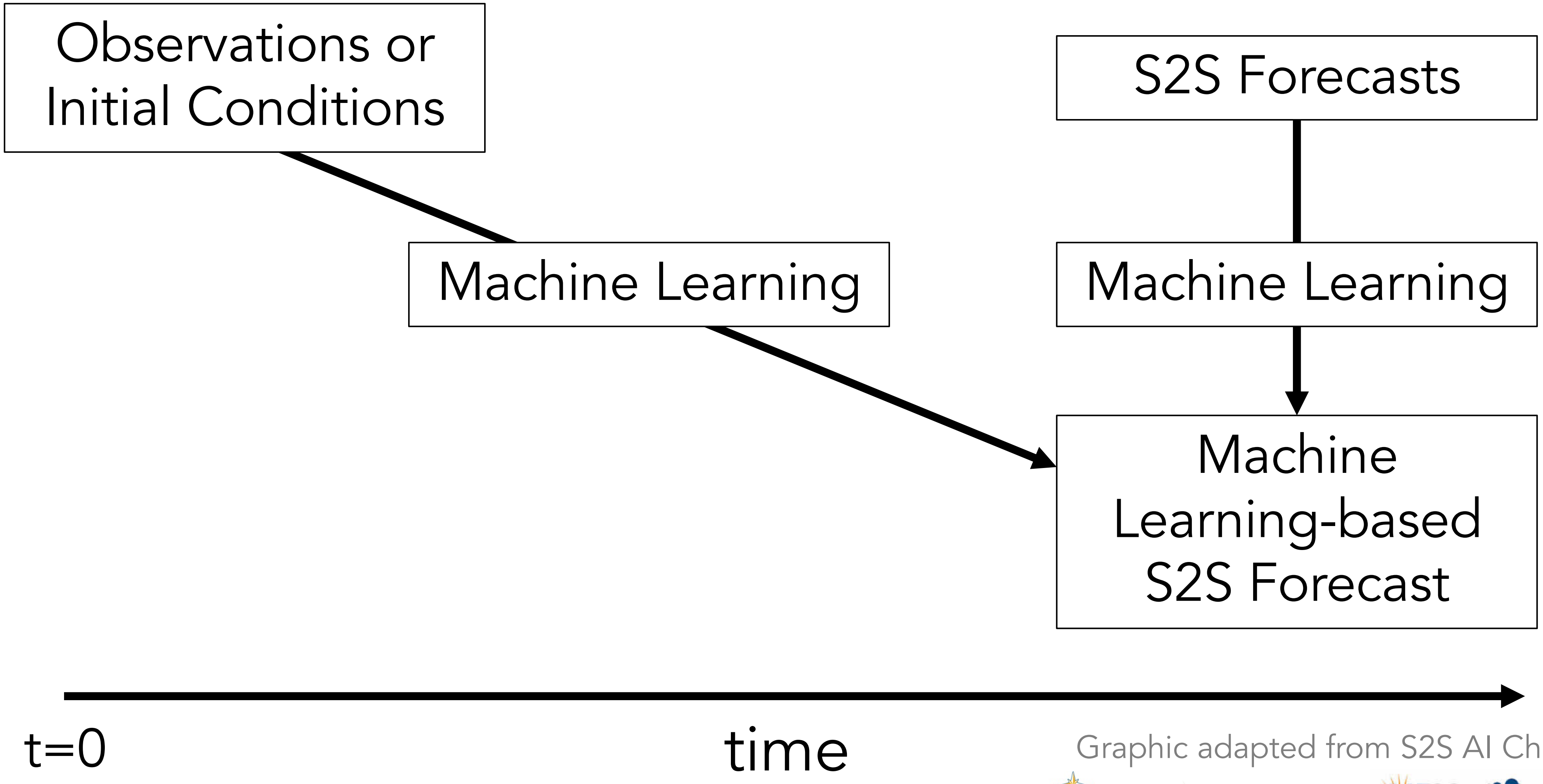
Temperature skill



Precipitation skill

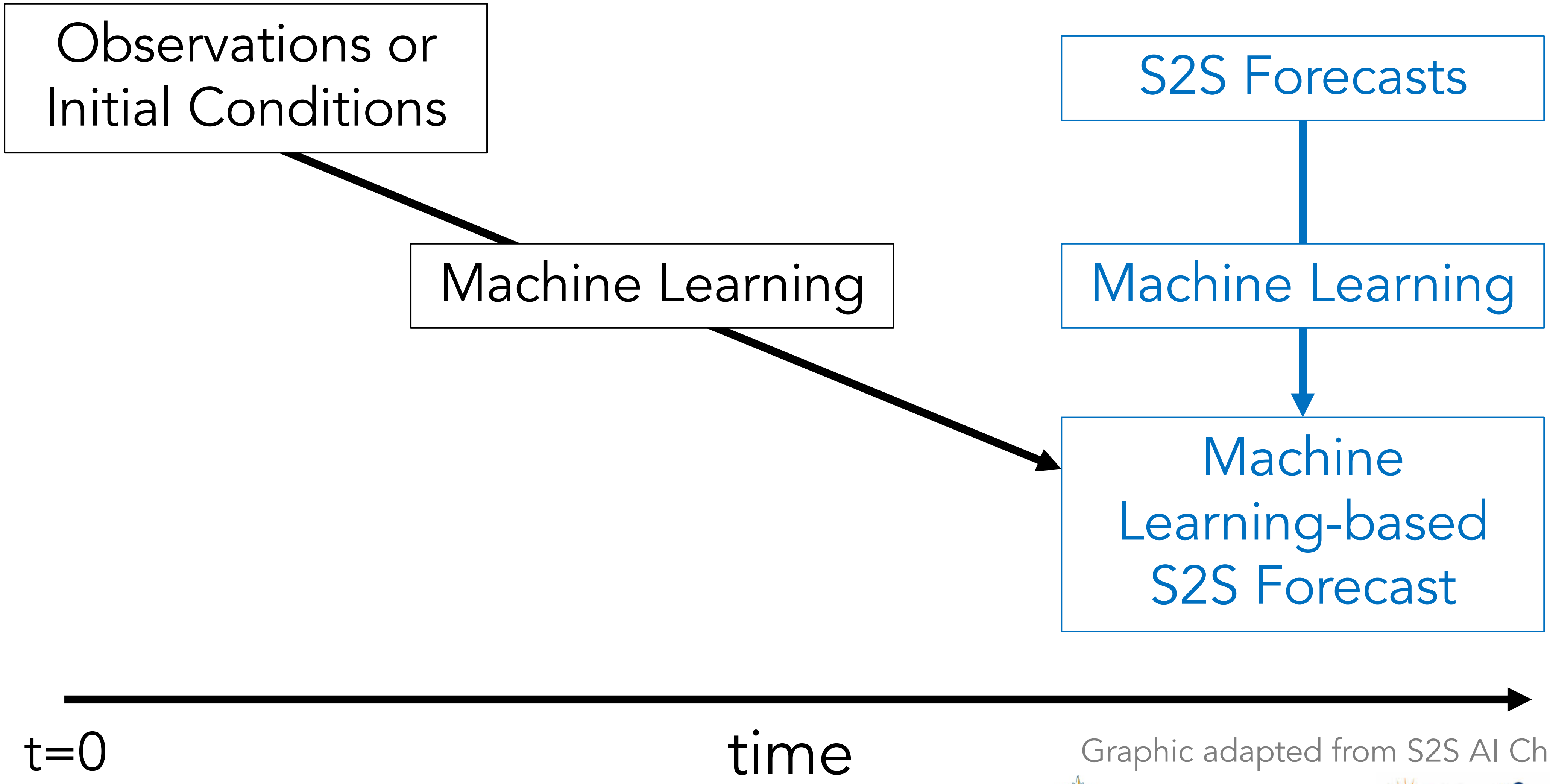


(Richter et al. 2022)



Graphic adapted from S2S AI Challenge 2021

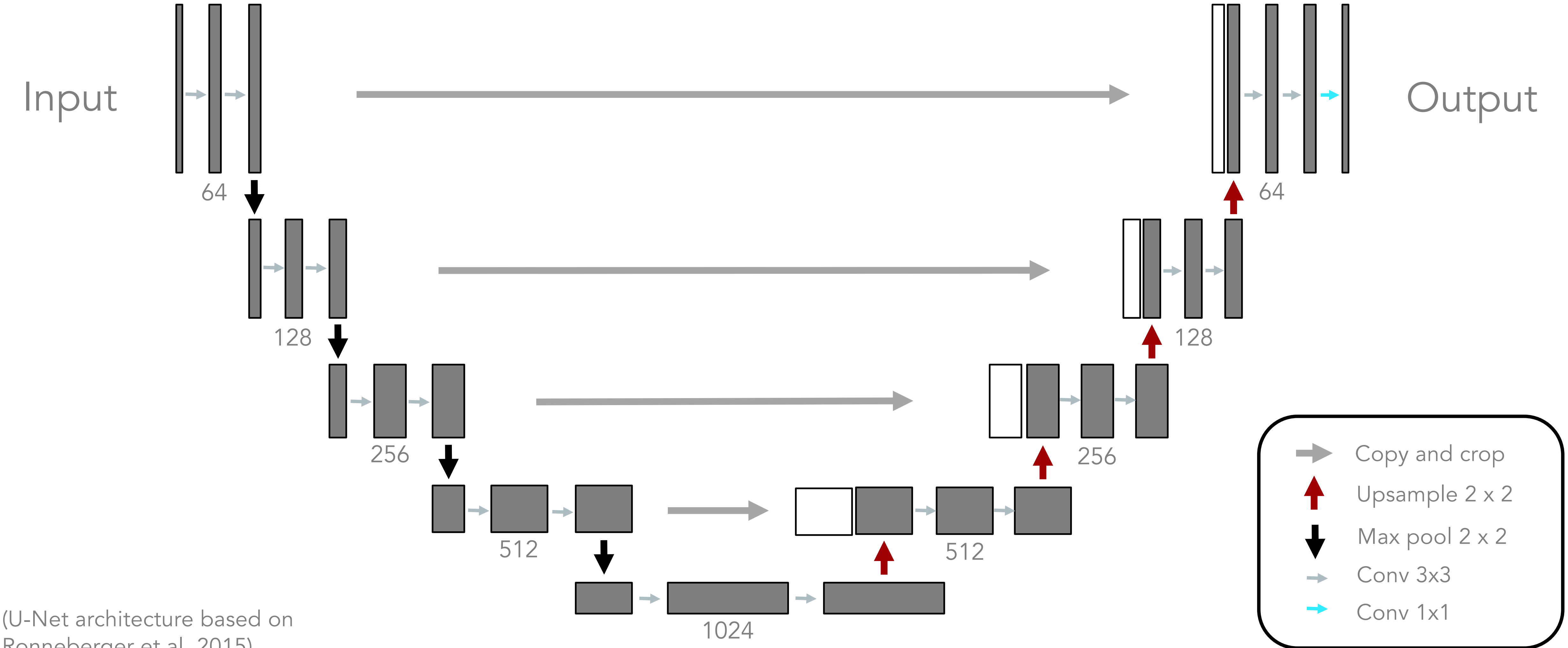
(Pegion et al. 2019, Merryfield et al. 2020, Barnes et al. 2020, Meehl et al. 2021)



Graphic adapted from S2S AI Challenge 2021

(Pegion et al. 2019, Merryfield et al. 2020, Barnes et al. 2020, Meehl et al. 2021)

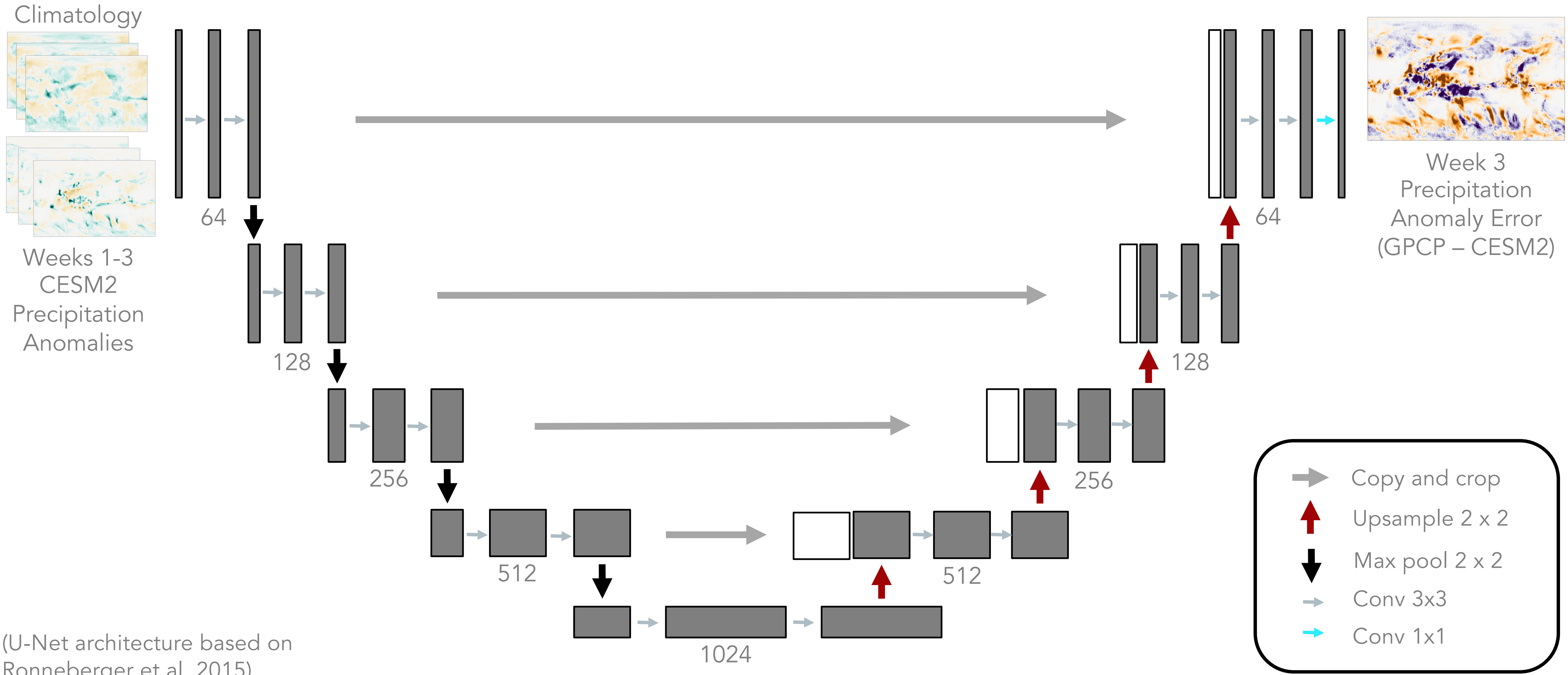
U-Net Architecture (training and validation: 1999-2015)



(U-Net architecture based on Ronneberger et al. 2015)

Molina et al. (in prep.)

U-Net Architecture (training and validation: 1999-2015)

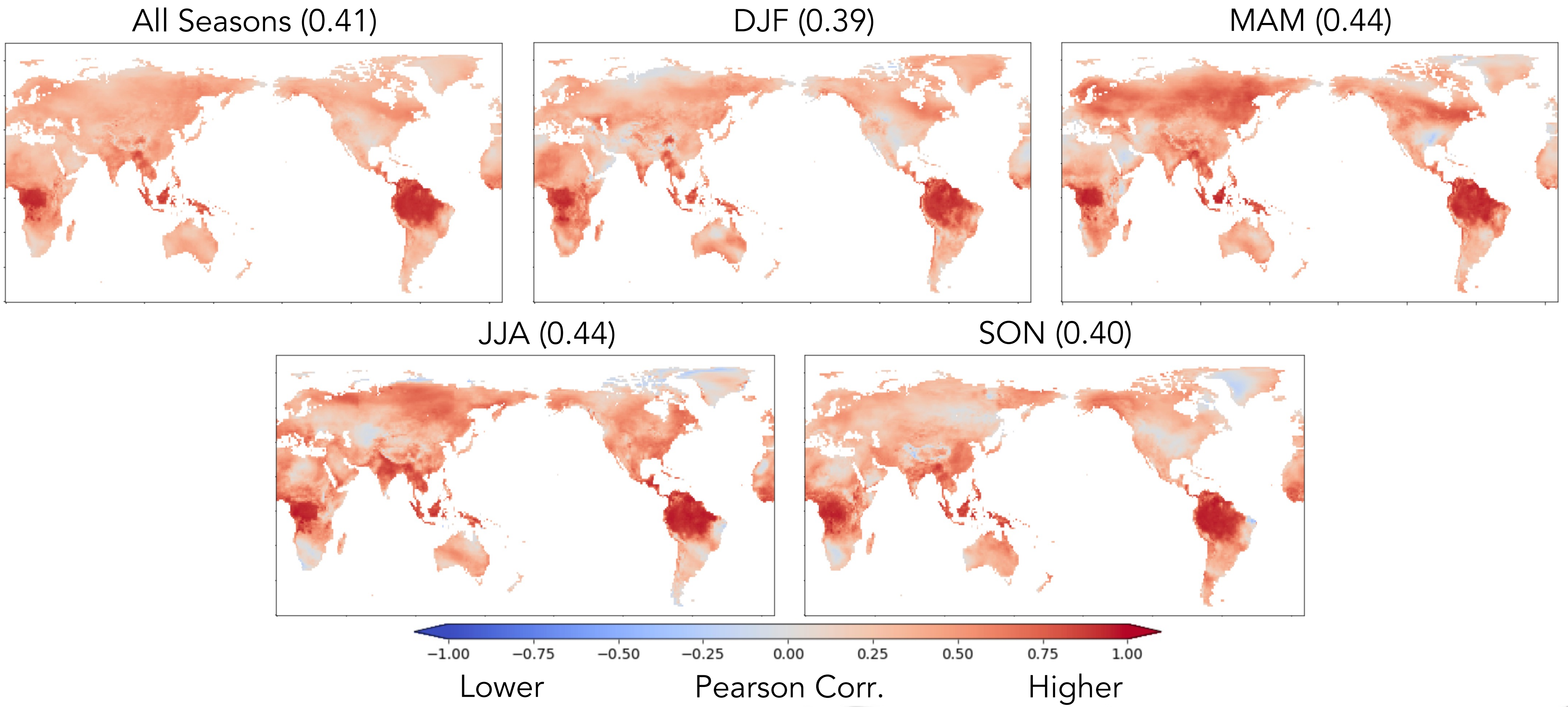


(U-Net architecture based on Ronneberger et al. 2015)

NCAR U.S. DEPARTMENT OF ENERGY Office of Science NSF

Molina et al. (in prep.)

Skill of Week 3 Temperature Error Prediction (2016-2019)



Molina et al. (in prep.)

Typical Workflow: (1) Data Preprocessing

Scientific Computing

Visualization

Typical Workflow: (2) Machine Learning

TensorFlow

scikit learn

ML / AI

PyTorch

K Keras

Typical Workflow: (3) ML Evaluation

xarray

pandas

Scientific Computing

MetPy

NumPy

DASK

SciPy

matplotlib

bokeh

Visualization

Cartopy

TensorFlow

scikit learn

seaborn

ML / AI

PyTorch

Keras

jupyterhub

Available NCAR Resources

Production

Development / Nightly

Subseasonal prediction of global temperature

Import packages

[1]:
```
import numpy as np
import xarray as xr
import pandas as pd
import matplotlib.pyplot as plt
from pylab import *
import xskillscore as xs
from IPython.display import Image, display

import torch
import torch.nn.functional as F
from torch.utils.data import DataLoader
import torch.nn as nn
import torch.optim as optim

import torch_funcs
import models
import torch_customdataset
import data_load
import models_unet
import models_unet2
```

network set up and training

```
[6]: net = models_unet.UNet(6,1)
```

```
[7]: LEARNING_RATE = 1e-4

# the optimizer
optimizer = optim.Adam(net.parameters(), lr=LEARNING_RATE, amsgrad=False)

# the loss function
criterion = nn.MSELoss(reduction='sum')
```

```
[8]: device = torch_funcs.get_device()
print(device)
net.to(device)
```

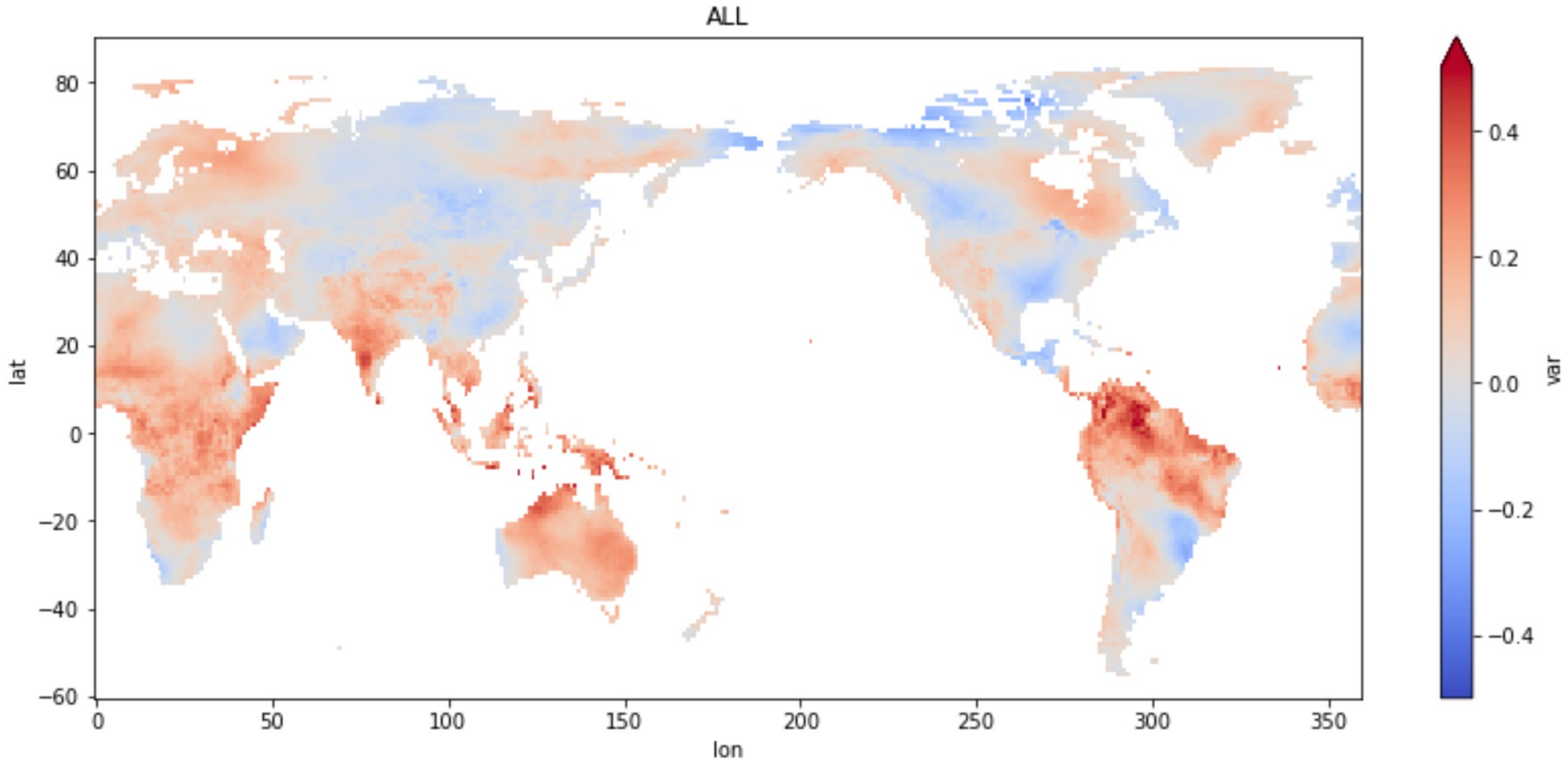
```
cuda:0
```

[9]:

```python
NUM_EPOCHS = 30

train_loss = []
valid_loss = []

train_corr = []
valid_corr = []

for enum, epoch in enumerate(range(NUM_EPOCHS)):

    t_loss, t_corr = train(net, train_loader, weights = None)
    v_loss, v_corr = validate(net, val_loader, weights = None)

    train_loss.append(t_loss)
    valid_loss.append(v_loss)

    train_corr.append(t_corr)
    valid_corr.append(v_corr)

    print(f"Epoch {epoch + 1} of {NUM_EPOCHS}; Train Loss: {t_loss:.4f}, Corr: {t_corr:.4f}; Val Loss: {v_los
```

```
Epoch 1 of 30; Train Loss: 7151747.6972, Corr: 0.4248; Val Loss: 6742937.4333, Corr: 0.4428
Epoch 2 of 30; Train Loss: 7000376.7007, Corr: 0.4453; Val Loss: 6677322.4333, Corr: 0.4528
Epoch 3 of 30; Train Loss: 6790247.8218, Corr: 0.4713; Val Loss: 6556439.9000, Corr: 0.4745
Epoch 4 of 30; Train Loss: 6334763.8287, Corr: 0.5236; Val Loss: 6141876.8000, Corr: 0.5247
Epoch 5 of 30; Train Loss: 5436897.0346, Corr: 0.6139; Val Loss: 5109534.9333, Corr: 0.6261
Epoch 6 of 30; Train Loss: 4380230.7569, Corr: 0.7057; Val Loss: 3985761.2167, Corr: 0.7235
Epoch 7 of 30; Train Loss: 3450515.8460, Corr: 0.7777; Val Loss: 3254654.8167, Corr: 0.7826
Epoch 8 of 30; Train Loss: 2764727.8564, Corr: 0.8270; Val Loss: 2717890.0833, Corr: 0.8229
Epoch 9 of 30; Train Loss: 2282221.7258, Corr: 0.8597; Val Loss: 2316539.0833, Corr: 0.8502
```

```python
acc = xs.pearson_r((obs).groupby('time.season')['SON'],     # skill for dynamical model (cesm)
                   (inp).groupby('time.season')['SON'],
                   dim='time', skipna=True)

print('CESM overall acc:',acc.weighted(torch_funcs.compute_lat_weights(obs)).mean(("lat", "lon")).values
```




```
ML overall acc: 0.07293882090150633
CESM overall acc: 0.10426163822977165
```

Key Software Needs

CONDA

NVIDIA

CUDA

CISL

github.com/mariajmolina

molina@ucar.edu

Me: *uses machine learning*

Machine: *learns*

Me: