Machine Learning-Informed Numerical Weather Prediction

Troy Arcomano    Istvan Szunyogh

Texas A&M University
College Station, TX

Supercomputing Conference (SC20), November 17-19th, 2020
General Approach

The **modeling approach** is based on **combining** (Pathak et al. 2018b and Wikner et al. 2020)

- a **numerical weather prediction (NWP) model** and
- a computationally highly efficient **machine learning (ML) algorithm** to obtain

- a **hybrid weather prediction (HWP) model** that provides more accurate predictions than either component

The **ML model component** uses

- a **parallel** (Pathak et al. 2018a) **reservoir computing** (Jaeger 2001, Maas et al. 2002, Lukoševicius and Jaeger 2009) approach

**Our goal** is to prove the concept by building a **low-resolution, global, HWP model**

Szunyogh and Arcomano  ML-Informed NWP  troyarcomano@tamu.edu
A ‘time step’ of the ML model is a composite function that predicts the physical state $u(t + \Delta t)$ from the physical state $u(t)$.

- **Input Layer**: Maps the physical state $u(t)$ into a much higher dimensional reservoir state $Wu(t)$ ($W$ is typically the matrix of a random projection).
- **Reservoir**: A high-dimensional dynamical system.
- **Output Layer**: Reads out the physical state $u(t + \Delta t)$ from the reservoir state $r(t + \Delta t)$. 

Szunyogh and Arcomano  
ML-Informed NWP  
troyarcomano@tamu.edu
Computationally Efficient, Parallel Algorithm

The global state vector $u(t)$ is partitioned into $L$ local state vectors:

- **Local State Vector**: Each local state vector is predicted independently (the linear regression problem is solved in parallel for the different local state vectors)

- **Extended Local State Vector**: Input layer operates on an extended local state vector, so information can propagate between the local regions
• **Simplified** Parameterizations, primitive **Equation DYnamics** Version 42 of the International Centre for Theoretical Physics (ICTP)
  • (Molteni 2003, Kucharski et al 2006)
• **Equations:**
  • Primitive equations
  • Simplified but modern parameterization
• **Resolution:**
  • 8 vertical layers
  • T30 (~300km)
• Been used to test and develop new numerical weather prediction and data assimilation techniques
Training

• Observation-based data set of past states of the atmosphere, regridded to SPEEDY horizontal and vertical grid

• Used the 5 prognostic variables for SPEEDY
  • Temperature
  • 2 components of the wind
  • Specific Humidity
  • Surface Pressure

• 11 years of data from 1981-1991
  • 9.5 years for training
  • 7 months for validation
Computational Details

- A distributed and parallel architecture
- Each local region is trained independently in parallel
  - Currently assigning 1 core per local region
  - 1152 regions used to represent the globe
- Dense and sparse linear algebra calculations are done using OpenMP threaded LAPACK, BLAS, and Sparse BLAS functions found in the Intel’s Math Kernel Library (MKL)
- Parallel IO
  - Non-collective, parallel HDF5 reading and writing of data
  - Reading in 750 GB of data with 1152 processors takes 10 minutes
- Real runtime for training over 10 years and making predictions using TAMU’s Ada cluster with 1152 cores and 2.8 Terabytes of total program memory is about 1 hour
• Comparing 20 hybrid forecasts to the regridded observation-based data not used for the training

• The 20 forecasts span from June 1990 to January 1991

• Forecast skill was compared to that of SPEEDY, persistence forecasts, and a reservoir computing based machine learning only model (trained using the same data as the hybrid)
A lower value of the root-mean-square error (RMSE) indicates a more accurate forecast.
Conclusion

• We built a prototype model that employs reservoir computing for ML-Informed numerical weather prediction (NWP)

• The hybrid system performs better than the numerical model out to 24 hours for all forecast variables
  • Atmospheric moisture and temperatures out to at least day 3

• Parallel IO can greatly improve runtime performance

• Reservoir computing algorithm with a parallel architecture allows for massively parallel training without a GPU, which is significantly faster than for a deep learning network