On the Application of a Neural Semi-Lagrangian Architecture for Weather Forecasting

Stéphane Gaudreault*[1], Carlos Pereira[1], Christopher Subich[1], Mohammad Mortezazadeh[1], Eldad Haber[2], David Millard[3], and Siqi Wei[1]

[1]Recherche en prévision numérique atmosphérique, Environnement et Changement climatique Canada, 2121 Route Transcanadienne, Dorval, H9P 1J3, QC, Canada
[2]Department of Earth, Ocean and Atmospheric Sciences, University of British Columbia, 2020 – 2207 Main Mall, Vancouver, V6T 1Z4, British Columbia, Canada
[3]Golisano College of CIS, Rochester Institute of Technology, Rochester, USA
stephane.gaudreault@ec.gc.ca

February 4, 2025

The current version of this work introduces the basis of the new Canadian PARADIS model for ML-based weather forecasting. Future versions of this manuscript will include the respective implementation details, results and analysis.

Abstract

Data-driven models have revolutionized weather forecasting. We present a physics-inspired neural network for weather prediction. Our approach mimics the advection-diffusion-reaction components of the physical mechanisms, allowing for a better representation of these phenomena with a significant improvement in the typical over-smoothing issue of existing ML-based weather forecasting. This method utilizes a semi-Lagrangian approach, where the interpolation and integration trajectories are part of the set of learned parameters.

1 From physical principles to a neural network

In meteorology, a Lagrangian form of the governing equations is often used to describe the motion of atmospheric properties by tracking individual air parcels as they move through space and time. At the coarsest level, atmospheric dynamics can be described by a system of differential equations of the form

$$\frac{D\mathbf{q}}{Dt} = \mathcal{F}(\mathbf{q}), \tag{1}$$

$$\frac{d\mathbf{x}}{dt} = \mathbf{V}, \tag{2}$$

where $\frac{D}{Dt} = \frac{\partial}{\partial t} + \mathbf{V} \cdot \nabla$ is the material derivative, $\mathbf{q}$ is the state vector, $\mathbf{x}$ is the position of a fluid element, $\mathbf{V}$ is a smooth velocity field and $\mathcal{F}$ is the function describing all forcings. This general ansatz encapsulates three fundamental types of physical processes

- Transport of fluid properties by winds (e.g., advection).
- Spatial mixing (e.g., pressure gradient forces and diffusion).

- Local interactions between variables (e.g., the Coriolis effect, cloud microphysics, and atmospheric chemistry).

This decomposition naturally suggests that an appropriate neural architecture would be one where different components specialize in capturing these types of physical processes.

In a neural network, it is often useful to design the architecture to work with a projection of the data into a latent space. Hence, starting with the input vector $\mathbf{q}(\mathbf{x}, t) = [\mathbf{q}_0(\mathbf{x}, t), \mathbf{q}_1(\mathbf{x}, t)], \ldots, \mathbf{q}_n(\mathbf{x}, t)]$, where m is the number of input channels, we first embed it in a higher-dimensional space to a vector $\mathbf{h}(\mathbf{x}, t)$ of $n > m$ channels. The previous ansatz, Eq.(**??**), is then reformulated as follows

$$\frac{D}{Dt}\mathbf{h}(\mathbf{x}, t) = \mathcal{F}_{\text{net}}(\mathbf{h}(\mathbf{x}, t)), \tag{3}$$
$$\frac{d\mathbf{x}}{dt} = \mathbf{V}_{\text{net}}.$$

Here, $\mathbf{h}(\mathbf{x}, t)$ represents the *hidden state* of the system at any time t, and $\mathcal{F}_{\text{net}}$ is a neural network that models the rate of change of the hidden state. Similarly, the velocity field $\mathbf{V}_{\text{net}} = [\mathbf{V}_1, \mathbf{V}_2]$ contains $2n$ channels, also implemented as a neural network. The architecture used to model these quantities can be chosen with a certain degree of freedom. For instance, they can be modeled using approaches such as convolutional neural networks (CNNs), transformers, or Kolmogorov-Arnold networks [**?**]. In this work, we choose a simple base architecture consisting of two layers of convolution with a kernel size of 3×3. The first convolution layer maintains the input dimension m and is followed by layer normalization and a SiLU (Swish) activation function. The second convolution maps from m to n channels while preserving spatial dimensions through appropriate padding. (See Appendix **??** for details on our padding approach). This block is consistently used for projection to and from the latent space, $\mathcal{F}_{\text{net}}$ and $\mathbf{V}_{\text{net}}$.

A common approach in numerical weather prediction for solving systems of the form (**??**) is the semi-Lagrangian method [**?**]. The semi-Lagrangian method offers several key advantages [**?**]: it is highly stable, allowing atmospheric models to use large time steps efficiently; it resolves phase speeds accurately; and it exhibits relatively low numerical dispersion.

The semi-Lagrangian method can be seen as a discrete version of the method of characteristics. First, it traces back the trajectory of a fluid parcel from its current position on the grid x^+ at time t^+, to its origin at the previous time step $t^- = t^+ - \Delta t$ by computing

$$x^- = x^+ - \Delta t \mathbf{V}_{\text{net}}. \tag{4}$$

Once the departure point x^- is determined, the method interpolates the field values h^- at these locations from the known values on the grid, as departure points generally does not align with the grid points. A common approach for this task is the Lagrange cubic interpolation [**?**]. Finally, one can integrate all the forcings and source terms along the traced trajectory of the parcel to obtain the values of h^+ at the next time step on the grid via

$$h^+ = h^- + \Delta t \mathcal{F}_{\text{net}}(h^-). \tag{5}$$

Appendix A Methods

A.1 Data Normalization

Normalization is a key step in preparing data for machine learning. It scales features to a similar range, making predictive models more accurate and reliable. Different normalization strategies are chosen based on the physical characteristics of the variables. For most atmospheric variables, such as temperature, wind components, and geopotential, we compute the mean and standard deviation independently at each pressure level. Then, Z-score normalization is applied as follows

$$q_{\text{normalized}} = \frac{q - \mu(q)}{\sigma(q)}. \tag{6}$$

Here, $\mu(q)$ and $\sigma(q)$ represent the mean and standard deviation of the variable q for a given pressure level. For precipitation, a logarithmic transformation is used to address its highly skewed distribution via

$$q_{\text{normalized}} = \log(q + \varepsilon) + c, \tag{7}$$

where ε is a small positive constant ensuring the argument of the logarithm remains strictly positive, and $c = 10$ is an offset. This transformation ensures non-negative values during inversion and effectively handles the skewness, where small values dominate over large ones.

Specific humidity requires a physically motivated logarithmic normalization

$$q_{\text{normalized}} = \frac{\log(\text{clip}(q, 0, q_{\max}) + \varepsilon) - \log(q_{\min})}{\log(q_{\max}) - \log(q_{\min})}, \tag{8}$$

where $q_{\min}$ and $q_{\max}$ are the global minimum and maximum humidity values in the dataset, and ε is a small constant. This method ensures non-negative normalized values and accounts for the large variation of specific humidity with altitude.

A.2 Spherical geometry

The implementation of the semi-Lagrangian scheme in spherical coordinates is inspired by [?], who introduced an "auxiliary spherical coordinate system" to address difficulties near the poles. For each arrival point (ϕ_a, λ_a), a rotated coordinate system (ϕ', λ') is defined, where the origin coincides with the arrival point, and the equator of this rotated system passes through the point in question. In this local system, λ' measures the angular distance along the rotated equator, while ϕ' measures the angular distance perpendicular to it.

The departure point coordinates are first computed in the rotated system

$$\begin{aligned} \lambda'_d &= -u_{\text{net}} \Delta t, \\ \phi'_d &= -v_{\text{net}} \Delta t, \end{aligned}$$

where u_{net} and v_{net} are the trained velocities obtained from the output of the V_{net} neural network, and Δt is the time step. The transformation from these rotated coordinates back to standard spherical coordinates is given by

$$\begin{aligned} \lambda &= \lambda_a + \text{arctan2}(\cos(\phi') \sin(\lambda'), \cos(\phi') \cos(\lambda') \cos(\phi_a) - \sin(\phi') \sin(\phi_a)), \\ \phi &= \arcsin\left(\sin(\phi') \cos(\phi_a) + \cos(\phi') \cos(\lambda') \sin(\phi_a)\right). \end{aligned}$$

Since the grid does not include the poles, a geocyclic padding similar to [?] is employed to handle boundary conditions for both convolution operations and trajectory interpolation. In the longitudinal direction, the domain is extended periodically, with points beyond $\lambda = 360°$ mapping to $\lambda = 0° +$remainder. For latitudinal boundaries, points beyond the poles are handled through reflection and rotation. For a point beyond $\phi = 90°$N, the padding coordinates are

$$\begin{aligned} \phi_{\text{pad}} &= 180° - \phi, \\ \lambda_{\text{pad}} &= \lambda + 180°. \end{aligned}$$

When applying the geocyclic padding, there is a difficulty in handling wind vectors near the Earth's poles due to the convergence of meridians, which creates artificial discontinuities in the latitude-longitude coordinate system. This would force the neural networks to learn unnatural, discontinuous transformations in their latent space representations. Hence, we propose to transform wind vectors from spherical coordinates into Cartesian coordinates in a pre-processing step. This transformation ensures that the wind components remain continuous across the poles. The transformation from spherical velocity components (u, v, w) to Cartesian components (u_x, u_y, u_z) is given by

$$\begin{aligned} u_x &= -u \sin(\lambda) - v \sin(\phi) \cos(\lambda) - w \cos(\phi) \cos(\lambda) \\ u_y &= u \cos(\lambda) - v \sin(\phi) \sin(\lambda) - w \cos(\phi) \sin(\lambda) \\ u_z &= v \cos(\phi) - w \sin(\phi). \end{aligned}$$

Then, the neural network operates on these continuous Cartesian components, and the output is transformed back to spherical coordinates during post-processing. The inverse transformation is given by

$$\begin{aligned} u &= -u_x \sin(\lambda) + u_y \cos(\lambda), \\ v &= -u_x \sin(\phi) \cos(\lambda) - u_y \sin(\phi) \sin(\lambda) + u_z \cos(\phi), \\ w &= -u_x \cos(\phi) \cos(\lambda) - u_y \cos(\phi) \sin(\lambda) - u_z \sin(\phi). \end{aligned}$$

A.3 Radiation Parametrization

Following the implementation of [?], we included the one-hour accumulated top-of-atmosphere incoming solar radiation as a conditioning channel available to the forecast model. We believe that the model uses this channel as a combined time-of-day and season-of-year signal, since it is not given enough information to compute a detailed radiative balance.

The calculation generally follows that of [?], and it takes into account the ellipticity of Earth's orbit but not the variable solar cycle. With λ as the local latitude, T as the Julian day (referenced to 1 Jan 2000, 12h UTC), $\delta(T)$ as the solar declination, $d(T)$ as the solar distance (in astronomical units), and $t(T, \lambda)$ as the longitude-dependent local solar time, the instantaneous top-of-atmosphere radiation is given by

$$R(\phi, \lambda, T) = \frac{1360.56}{d(T)^2} \max(0, \sin(\phi) \sin(\delta(T)) + \cos(\phi) \cos(\delta(T)) \cos(t(T, \lambda))), \tag{9}$$

which has units of watts per square meter.

The solar distance, declination, and local solar time depend on the parameters of the Earth's orbit and the Julian day. The time-dependent obliquity (in radians) of Earth's orbit is approximately

$$\epsilon(T) = \frac{\pi}{180}(23.439 - 3.6 \cdot 10^{-7} T),$$

the mean anomaly is

$$M(T) = \frac{\pi}{180}(357.529 + 0.985600028T),$$

the mean longitude is

$$L(T) = \frac{\pi}{180}(280.459 + 0.98564736T),$$

the geocentric apparent longitude is

$$\lambda_{\odot}(T) = L(T) + \frac{\pi}{180}(1.915 \sin(M(T)) + 0.020 \sin(2M(T))),$$

the solar distance is

$$d(T) = 1.00014 - 0.01671 \cos(M(T)) - 1.4 \cdot 10^{-4} \cos(2M(T)),$$

the right ascension is

$$\alpha(T) = \arctan\left(\frac{\cos(\epsilon(T)) \sin(\lambda_{\odot}(T)}{\cos(\lambda_{\odot}(T)}\right),$$

and the declination is

$$\delta(T) = \arcsin(\sin(\epsilon(T)) \sin(\lambda_{\odot}(T))).$$

Computing the local solar time requires computing the equation of time

$$EOT(T) = L(T) - \alpha(T),$$

and finally, the local solar time is obtained via

$$t(T, \lambda) = \lambda + 2\pi T + EOT(T).$$

During computation, all radian values are limited to the $[-\pi, \pi]$ interval, and the solar radiation is accumulated by integrating (**??**) over the 1-hour period that ends at the specified time.

Appendix B Code availability

The implementation of the PARADIS model presented in this work is available on GitHub at https://github.com/Wx-Alliance-Alliance-Meteo/paradis_model.

References

[1] Ziming Liu, Yixuan Wang, Sachin Vaidya, Fabian Ruehle, James Halverson, Marin Soljačić, Thomas Y Hou, and Max Tegmark. KAN: Kolmogorov–arnold networks. In *Submitted to The Thirteenth International Conference on Learning Representations*, 2024. under review.

[2] André Robert. A stable numerical integration scheme for the primitive meteorological equations. *Atmosphere-Ocean*, 19(1):35–46, 1981.

[3] Steven J Fletcher. *Semi-Lagrangian advection methods and their applications in geoscience*. Elsevier, 2019.

[4] A McDonald. Accuracy of multiply-upstream, semi-lagrangian advective schemes. *Monthly Weather Review*, 112(6):1267–1275, 1984.

[5] A McDonald and JR Bates. Semi-lagrangian integration of a gridpoint shallow water model on the sphere. *Monthly Weather Review*, 117(1):130–137, 1989.

[6] Minjong Cheon, Yo-Hwan Choi, Seon-Yu Kang, Yumi Choi, Jeong-Gil Lee, and Daehyun Kang. Karina: An efficient deep learning model for global weather forecast. *arXiv preprint arXiv:2403.10555*, 2024.

[7] Remi Lam, Alvaro Sanchez-Gonzalez, Matthew Willson, Peter Wirnsberger, Meire Fortunato, Ferran Alet, Suman Ravuri, Timo Ewalds, Zach Eaton-Rosen, Weihua Hu, et al. Learning skillful medium-range global weather forecasting. *Science*, 382(6677):1416–1421, 2023.

[8] Lucien Wald. Basics in solar radiation at Earth surface v2, July 2019.