Learning to forecast diagnostic parameters using pre-trained weather embedding

Peetak Mitra
Excarta
peetak@excarta.io

Vivek Ramavajjala
Excarta
vivek@excarta.io

Abstract

Data-driven weather prediction (DDWP) models are increasingly becoming popular for weather forecasting. However, while operational weather forecasts predict a wide variety of weather variables, DDWPs currently forecast a specific set of key prognostic variables. Non-prognostic ("diagnostic") variables are sometimes modeled separately as dependent variables of the prognostic variables (c.f. FourCastNet [10]), or by including the diagnostic variable as a target in the DDWP. However, the cost of training and deploying bespoke models for each diagnostic variable can increase dramatically with more diagnostic variables, and limit the operational use of such models. Likewise, retraining an entire DDWP each time a new diagnostic variable is added is also cost-prohibitive. We present an two-stage approach that allows new diagnostic variables to be added to an end-to-end DDWP model without the expensive retraining. In the first stage, we train an autoencoder that learns to embed prognostic variables into a latent space. In the second stage, the autoencoder is frozen and "downstream" models are trained to predict diagnostic variables using only the latent representations of prognostic variables as input. Our experiments indicate that models trained using the two-stage approach offer accuracy comparable to training bespoke models, while leading to significant reduction in resource utilization during training and inference. This approach allows for new "downstream" models to be developed as needed, without affecting existing models and thus reducing the friction in operationalizing new models.

1 Introduction

In recent years, data-driven weather prediction (DDWP) models, such as FourCastNet [10], GraphCast [5], and PanguWeather [1], have shown remarkable skill in multi-day forecasts compared to state-of-the-art, operational Numerical Weather Prediction (NWP) models such as Integrated Forecast Service (IFS) [12], and reanalyses data such as ERA5 [9]. Recent studies have indicated that these DDWP models and NWP emulators, while fully data-dependent, have learnt useful physics proving the robustness of this approach [2], and behave similarly to NWPs when comparing against *in-situ* observations [11]. However, most DDWPs often do not focus on diagnostic meteorological parameters such as precipitation, total cloud cover, solar irradiance, soil moisture, etc., that are essential for decision making in industries like transportation, energy, and agriculture. GraphCast[5] forecasts precipitation along with other prognostic variables, while FourCastNet[10] trains a bespoke model to predict precipitation, with the same complexity as the "backbone" model itself (Figure 1). In either approach, a full-scale DDWP needs to be re-trained for each new diagnostic variable to be predicted, which is not scalable either in training or in inference.

It is commonly observed that the first few layers of most deep learning models learn a dense latent representation of the high-dimensional input space[7, 13, 14]. Since in most cases, the starting state for predict diagnostic variables is the current set of prognostic variables, we propose a two-stage approach

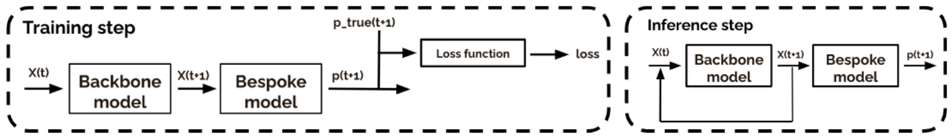

Figure 1: FourCastNet used a bespoke model to predict precipitation. A backbone model that only predicts "core" weather state, and a bespoke model that uses the weather state to predict diagnostic variables. Scaling such an approach for many downstream tasks would be challenging.

to modeling diagnostic variables: first, we train an auto-encoder that learns to embed prognostic variables in a dense latent space. Then, for each diagnostic variable we train a "downstream" model that uses the dense representation of prognostic variables as an input. As the auto-encoder has already learned a high-quality dense representation, we can significantly reduce the size of each "downstream" model. The closest such approach is proposed in W-MAE [8], which also trains an auto-encoder but fine-tunes the auto-encoder to forecast prognostic variables and precipitation. Fine tuning the encoder prevents new diagnostic variables from being modeled without also re-training all previous models.

In contrast, we propose training an autoencoder using decades of high quality ground truth hourly re-analyses data from ERA5, which is then frozen, and using the frozen auto-encoder to train downstream tasks. The encoder's dense representations can be used as-is, or be combined with other task-specific signals to directly forecast industry-specific weather-dependent outcomes, e.g., predicting likelihood of pest outbreaks as a function of humid weather conditions.

1.1 Our contributions

The main contributions of this study are in demonstrating that (i) complex, high-dimensional weather data can be effectively represented in a condensed, embedded representation at a moderate cost, and (ii) weather-dependent tasks can be learned efficiently from dense representations of weather. We demonstrate this by not only quantifying the reconstruction error of the autoencoder model, but using it to build task-specific downstream applications, and comparing their performance to bespoke models. This finding unlocks the ability to readily use the weather state in key decision making processes by dramatically reducing the cost of training and deploying ML-driven weather models.

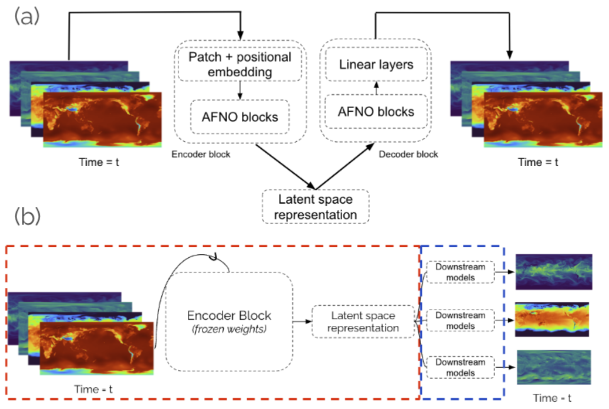

Figure 2: **a**: The autoencoder model learns, via reconstructing the input, a rich dense representation of the current weather state, and **b**: The trained encoder is frozen and used to produce latent representations of the current weather state (block in *red*), which are used as inputs to train smaller downstream branches (block in *blue*).

2 Experimental design

Our objective is to demonstrate that the two-stage modeling approach described above provides performance comparable to training bespoke, full-scale DDWPs for diagnostic variables.

2.1 Dataset

The dataset used for this study is the publicly available reanalyses data product, ERA5 [3]. We choose a subset of the variables (defined in Table 1) provided in ERA5 which are commonly considered to be prognostic variables, and thus critical to capturing the essential information about the current weather state. These variables are then normalized to zero mean and unit variance using climatological statistics computed over decades of data. In our experiments, we focus on modeling total cloud cover (tcc), and top-level soil temperature ($\mathrm{stl1}$), which are critical for operational decisions in the energy and agriculture industry, respectively. While our approach was demonstrated on these variables for this study, this approach is task-agnostic and can readily be extended to model any weather or non-weather variables.

2.1.1 Input and target variables

The 54 prognostic input variables, used at 0.25°resolution, are similar to the parameters used in other DDWP studies [10, 1].

Table 1: Model inputs, totalling 54 weather variables

Vertical Level	Variables
Surface	$\mathrm{u_{10}, v_{10}, t_{2m}, d_{2m}, msl, sp, u_{100}, v_{100}, tcwv}$
1000, 925, 850, 700, 500, 300, 250, 200, 50 (all in hPa)	$\mathrm{t, u, v, z, r}$

The target variables are listed in Table 2. While the modeling approach is variable-agnostic, we choose these two variables because of their relative importance across the energy and agriculture industries, and also their relevant modeling challenges. For example, total cloud cover (*tcc*) is influenced by conditions at all levels of the atmosphere, while top-level soil temperature (*stl1*) is influenced more heavily by surface weather. For evaluation, we use the RMSE metric for both total cloud cover and soil temperature. For total cloud cover, we additionally use the structural similarity metric (SSIM) to measure the spatial similarity in the predicted and true cloud cover.

Table 2: Target variables used in the study

Variable	Data Range	Activation	Masks
$\mathrm{stl1}$	220-290 K	None	land-sea
tcc	0-1	softmax	None

2.2 Model architecture

Our two-stage training approach is as follows:

1. Train an auto-encoder on ERA5 data to learn a dense representation of the prognostic variables at time *t*
2. Train task-specific models to predict diagnostic variables using the dense representation as input.

Since only the encoder model sees the prognostic variables, and the encoder does not make assumptions about the prognostic variables, the above setup can be coupled with any "backbone" forecasting models as long as the necessary prognostic variables are available. In other words, this approach can be coupled with GraphCast, Pangu-Weather, or any DDWP, to automatically predict additional diagnostic variables.

2.3 Autoencoder

The weather state embedding is built using an autoencoder which consists of two different modules, an encoder (ϕ) that represents the input data $[\mathrm{B}, \mathrm{C_{in}}, \mathrm{H_{in}}, \mathrm{W_{in}}]$ into a dense, latent representation

$[\mathrm{B}, \mathrm{L}, \mathrm{H_L}, \mathrm{W_L}]$, functionally shown as

$$\phi : \chi_t \rightarrow F \tag{1}$$

and a decoder (ψ) which uses the latent representation, to recreate the original data $[\mathrm{B}, \mathrm{C_{out}}, \mathrm{H_{out}}, \mathrm{W_{out}}]$, functionally represented as

$$\psi : F \rightarrow \chi_t \tag{2}$$

During training, the overall minimization process updates the encoder and decoder weights simultaneously, until fully trained.

$$\phi, \psi = \texttt{arg min}_{\phi,\psi} \|\chi_t - (\phi \circ \psi)\chi_t\|^2 \tag{3}$$

Once trained, only the encoder is used to embed the current weather state into a dense representation. In our study, we use construct the autoencoder using AFNO layers as described in FourCastNet [10]. However, this approach is architecture-agnostic, meaning other architectures like Vision Transformers (ViT) or graph-based methods can similarly be used. Table 3 defines the architecture choices and total model sizes.

2.4 Task-specific downstream model

The task specific downstream models (M) use the dense representations from the auto-encoder as the input, functionally seen as $\mathrm{Y_t} = \mathrm{M}(\phi(\chi_\mathrm{t}))$.

2.5 Bespoke models

To enable a fair comparison of the performance of the task-specific downstream model, we train bespoke models based on the FourCastNet backbone model architecture [10], using the same prognostic variables as input (c.f. Table 1) and the same diagnostic variables (c.f. Table 2) as output. All models use standard Adam [4] optimizer with an exponential learning rate decay of an initial rate of 0.0002.

The following table shows the model sizes used in the study, and we note that the "downstream" models are significantly smaller than the bespoke models.

Table 3: Model architecture choices and total number of parameters

	AFNO Hyperparameters			
Model	# of layers	dim	patch size	Total # of parameters
Autoencoder	4 each for encoder, decoder	768	8	49M
Downstream models	6	768	8	28M
Bespoke models	12	768	8	75M

3 Evaluations and Discussions

The evaluations were performed over the hold-out test year of 2020. To holistically characterize the model performance and account for seasonality, we evaluate the performance on the 1st, 2nd, 15th and 16th days of each month. For each date, evaluations are performed for each hour of the day to account for the time-of-day effect. The forecasts are evaluated by comparing the predictions against ERA5 data using two metrics: the root mean squared error (RMSE) and the structural similarity index measure (SSIM), computed hourly over the entire latitude-longitude grid. The mean and standard deviation are computed by collating the data from across all evaluation periods. While the RMSE provides a global average of error, the SSIM metric is a qualitative, reconstructive, perceptual metric that quantifies image quality degradation to the ground truth data.

3.1 Autoencoder

Autoencoders often produce manifolds that over fit to noisy training data [6]. Such over fitting could lead to a poor starting state for the task-specific fine tuning models degrading their performance.

To prevent overfitting and understand the impact of model architecture choices, we performed ablation studies on the patch size, number of AFNO layers, and the number of channels used in the autoencoder. Results indicate that reducing the patch size leads to slight improvement in performance, but the training and inference cost scales quadratically for when doubling patch sizes, leading to a significantly unfavorable compute-accuracy trade off for smaller patch sizes. For this study, we use the encoder with a patch size of 8, given its lower computational overhead and overall similar performance to the patch size 4 models. The number of AFNO layers and channels are by far the most important hyperparameters (c.f. Table 4), in line with the expectations that an over-parameterized model can more easily overfit.

Table 4: Ablation study for autoencoder model

Patch size, # Layers	variable	Avg. RMSE: μ, σ	Avg. SSIM: μ, σ
patch:4, layers:8	u10	0.187, 0.0023	0.9947, 0.0004
patch:4, layers:4	u10	**0.128, 0.0015**	**0.9973, 0.0002**
patch:8, layers:8	u10	0.185, 0.0010	0.9937, 0.0005
patch:8, layers:4	u10	0.134, 0.0016	0.9947, 0.0004
patch:4, layers:8	t2m	0.676, 0.0344	0.9920, 0.0005
patch:4, layers:4	t2m	0.481, 0.0221	**0.9944, 0.0004**
patch:8, layers:8	t2m	0.581, 0.0114	0.9877, 0.0007
patch:8, layers:4	t2m	**0.454, 0.0105**	0.9886, 0.0007

3.2 Inter model comparisons

The starting point for the task-specific downstream model is the dense representation from the trained encoder model. In the table 5, we compare the average RMSE and SSIM scores for the two modeled variables between the two separate modeling approaches. Results indicate that the average errors between the bespoke and "downstream" models are almost nearly identical, further validating the two-stage modeling approach. However, each downstream model is roughly half the size of the corresponding bespoke models, and during inference the auto-encoder needs to be run only once for any number of downstream tasks. The significantly lower overhead during training, and dramatically lower footprint during inference, makes the cost-accuracy trade off hugely favorable for the smaller downstream models.

Table 5: Comparing model performance on diagnostic variables

Model type	variable	Avg. RMSE: μ, σ	Avg. SSIM: μ, σ
Bespoke	tcc	**0.1656, 0.0076**	0.5648, 0.0134
Downstream	tcc	0.1677, 0.0084	**0.5926, 0.0141**
Bespoke	stl1	**11.761, 0.14367**	**0.9633, 0.00034**
Downstream	stl1	11.784, 0.1645	0.9632, 0.00036

4 Conclusions

While DDWPs have steadily improved their forecast skill, their use in operational forecasts is limited if only prognostic variables are predicted. Many decisions in the real world are dependent on diagnostic variables, and neither retraining DDWPs nor building bespoke models for each new diagnostic variables are scalable. The main impetus of this work is to develop efficient, scalable ways to train and operationalize new models dependent on weather. We show that prognostic variables can be embedded in a task-agnostic latent space, and be used as a starting state to train task-specific downstream models without any appreciable drop in model accuracy, and significant savings in computational costs.

5 Data and code availability

The authors are working to release an extended paper with more evaluations and open source the trained model weights. In the meantime, if you have any questions please reach out to us at contact@excarta.io.

References

[1] Kaifeng Bi, Lingxi Xie, Hengheng Zhang, Xin Chen, Xiaotao Gu, and Qi Tian. Accurate medium-range global weather forecasting with 3d neural networks. *Nature*, pages 1–6, 2023.

[2] Gregory J Hakim and Sanjit Masanam. Dynamical tests of a deep-learning weather prediction model. *arXiv preprint arXiv:2309.10867*, 2023.

[3] Hans Hersbach, Bill Bell, Paul Berrisford, Shoji Hirahara, András Horányi, Joaquín Muñoz-Sabater, Julien Nicolas, Carole Peubey, Raluca Radu, Dinand Schepers, et al. The ERA5 global reanalysis. *Quarterly Journal of the Royal Meteorological Society*, 146(730):1999–2049, 2020.

[4] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[5] Remi Lam, Alvaro Sanchez-Gonzalez, Matthew Willson, Peter Wirnsberger, Meire Fortunato, Alexander Pritzel, Suman Ravuri, Timo Ewalds, Ferran Alet, Zach Eaton-Rosen, Weihua Hu, Alexander Merose, Stephan Hoyer, George Holland, Jacklynn Stott, Oriol Vinyals, Shakir Mohamed, and Peter Battaglia. Graphcast: Learning skillful medium-range global weather forecasting, 2022.

[6] Yonghyeon Lee, Hyeokjun Kwon, and Frank Park. Neighborhood reconstructing autoencoders. *Advances in Neural Information Processing Systems*, 34:536–546, 2021.

[7] Yuchen Li, Yuanzhi Li, and Andrej Risteski. How do transformers learn topic structure: Towards a mechanistic understanding. *arXiv preprint arXiv:2303.04245*, 2023.

[8] Xin Man, Chenghong Zhang, Changyu Li, and Jie Shao. W-mae: Pre-trained weather model with masked autoencoder for multi-variable weather forecasting. *arXiv preprint arXiv:2304.08754*, 2023.

[9] Tung Nguyen, Johannes Brandstetter, Ashish Kapoor, Jayesh K Gupta, and Aditya Grover. Climax: A foundation model for weather and climate. *arXiv preprint arXiv:2301.10343*, 2023.

[10] Jaideep Pathak, Shashank Subramanian, Peter Harrington, Sanjeev Raja, Ashesh Chattopadhyay, Morteza Mardani, Thorsten Kurth, David Hall, Zongyi Li, Kamyar Azizzadenesheli, Pedram Hassanzadeh, Karthik Kashinath, and Animashree Anandkumar. FourCastNet: A Global Data-driven High-resolution Weather Model using Adaptive Fourier Neural Operators, 2022.

[11] Vivek Ramavajjala and Peetak P Mitra. Verification against in-situ observations for data-driven weather prediction. *arXiv preprint arXiv:2305.00048*, 2023.

[12] Stephan Rasp, Stephan Hoyer, Alexander Merose, Ian Langmore, Peter Battaglia, Tyler Russel, Alvaro Sanchez-Gonzalez, Vivian Yang, Rob Carver, Shreya Agrawal, et al. Weatherbench 2: A benchmark for the next generation of data-driven global weather models. *arXiv preprint arXiv:2308.15560*, 2023.

[13] Matthew D Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. In *Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part I 13*, pages 818–833. Springer, 2014.

[14] Daquan Zhou, Zhiding Yu, Enze Xie, Chaowei Xiao, Animashree Anandkumar, Jiashi Feng, and Jose M Alvarez. Understanding the robustness in vision transformers. In *International Conference on Machine Learning*, pages 27378–27394. PMLR, 2022.