Learning skillful medium-range global weather forecasting

Remi Lam,[1*] Alvaro Sanchez-Gonzalez,[1*] Matthew Willson,[1*] Peter Wirnsberger,[1*] Meire Fortunato,[1*] Ferran Alet,[1*] Suman Ravuri,[1*] Timo Ewalds,[1] Zach Eaton-Rosen,[1] Weihua Hu,[1] Alexander Merose,[2] Stephan Hoyer,[2] George Holland,[1] Oriol Vinyals,[1] Jacklynn Stott,[1] Alexander Pritzel,[1] Shakir Mohamed,[1] Peter Battaglia[1]

[1]Google DeepMind

[2]Google Research

[*]Equal contributions.

Global medium-range weather forecasting is critical to decision-making across many social and economic domains. Traditional numerical weather prediction uses increased compute resources to improve forecast accuracy, but does not directly use historical weather data to improve the underlying model. Here, we introduce "GraphCast", a machine learning-based method trained directly from reanalysis data. It predicts hundreds of weather variables, over 10 days at 0.25° resolution globally, in under one minute. GraphCast significantly outperforms the most accurate operational deterministic systems on 90% of 1380 verification targets, and its forecasts support better severe event prediction, including tropical cyclones tracking, atmospheric rivers, and extreme temperatures. GraphCast is a key advance in accurate and efficient weather forecasting, and helps realize the promise of machine learning for modeling complex dynamical systems.

It is 05:45 UTC in mid-October, 2022, in Bologna, Italy, at the European Centre for Medium-Range Weather Forecasts (ECMWF)'s new High-Performance Computing Facility, which recently opened for operation. For the past several hours the Integrated Forecasting System (IFS) has been running sophisticated calculations to forecast Earth's weather over the next days and weeks, and its first predictions have just begun to be disseminated to users. This process repeats every six hours, every day, to supply the world with the most accurate weather forecasts available.

The IFS, and modern weather forecasting more generally, are triumphs of science and engineering. The dynamics of weather systems are among the most complex physical phenomena on Earth, and each day, countless decisions made by individuals, industries, and policymakers depend on accurate weather forecasts, from deciding whether to wear a jacket or to flee a dangerous storm. The dominant approach for weather forecasting today is "numerical weather prediction" (NWP), which involves solving the governing equations of weather using supercomputers. The success of NWP lies in the rigorous and ongoing research practices that provide increasingly detailed descriptions of weather phenomena, and how well NWP scales to greater accuracy with greater computational resources (*1, 2*). As a result, the accuracy of weather forecasts has increased year after year, to the point where the path of a hurricane can be predicted many days ahead—a possibility that was unthinkable even a few decades ago.

But while traditional NWP scales well with compute, capitalizing on the vast amount of historical weather data to improve accuracy is not straightforward. Rather, NWP methods are improved by highly trained experts innovating better models, algorithms, and approximations, which can be a time-consuming and costly process.

Machine learning-based weather prediction (MLWP) offers an alternative to traditional NWP, where forecast models can be trained from historical data, including observations and analysis data. This has potential to improve forecast accuracy by capturing patterns in the data

which are not easily represented in explicit equations. MLWP also offers opportunities for greater efficiency by exploiting modern deep learning hardware, rather than supercomputers, and striking more favorable speed-accuracy trade-offs. Recently MLWP has helped improve on NWP-based forecasting in regimes where traditional NWP is relatively weak, for example sub-seasonal heat wave prediction (*3*) and precipitation nowcasting from radar images (*4–7*), where accurate equations and robust numerical methods are not as available.

In medium-range weather forecasting, i.e., predicting atmospheric variables up to 10 days ahead, NWP-based systems like the IFS are still most accurate. The top deterministic operational system in the world is ECMWF's High RESolution forecast (HRES), a configuration of IFS which produces global 10-day forecasts at 0.1° latitude/longitude resolution, in around an hour (*8*). However, over the past several years, MLWP methods for medium-range forecasting trained on reanalysis data have been steadily advancing, facilitated by benchmarks such as WeatherBench (*8*). Deep learning architectures based on convolutional neural networks (*9–11*) and Transformers (*12*) have shown promising results at latitude/longitude resolutions coarser than 1.0°, and recent works—which use graph neural networks (GNN), Fourier neural operators, and Transformers (*13–16*)—have reported performance that begins to rival IFS's at 1.0° and 0.25° for a handful of variables, and lead times up to seven days.

Surface variables (5)	Atmospheric variables (6)	Pressure levels (37)
2-meter temperature (2T)	**Temperature** (T)	1, 2, 3, 5, 7, 10, 20, 30, **50**, 70,
10 metre u wind component (10U)	**U component of wind** (U)	**100**, 125, **150**, 175, **200**, 225,
10 metre v wind component (10V)	**V component of wind** (V)	**250**, **300**, 350, **400**, 450, **500**,
Mean sea-level pressure (MSL)	**Geopotential** (Z)	550, **600**, 650, **700**, 750, 775,
Total precipitation (TP)	**Specific humidity** (Q)	800, 825, **850**, 875, 900, **925**,
	Vertical wind speed (W)	950, 975, **1000**

Table 1: **Weather variables and levels modeled by GraphCast.** The numbers in parentheses in the column headings are the number of entries in the column. Boldfaced variables and levels indicates those which were included in the scorecard evaluation. All atmospheric variables are represented at each of the pressure levels.

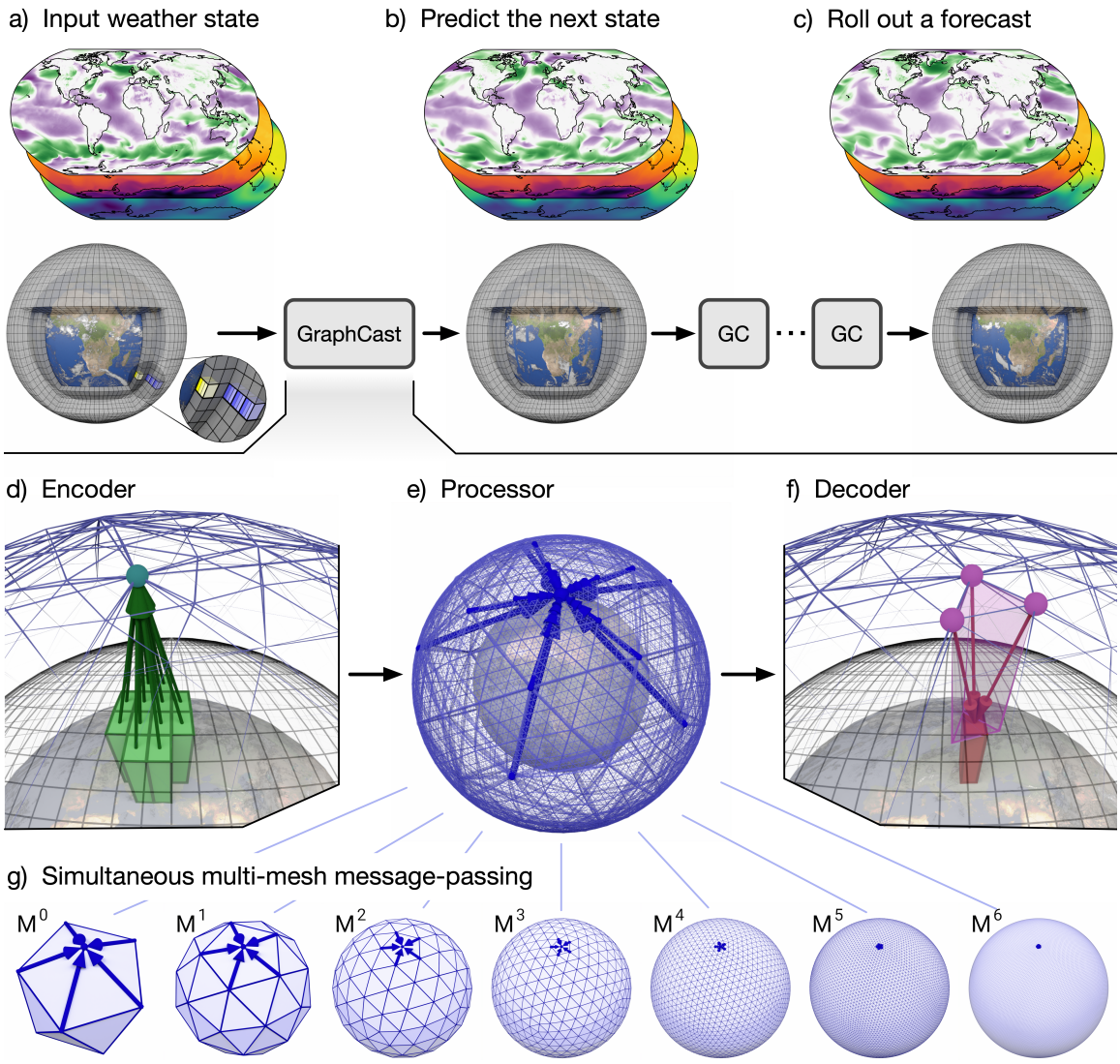

Fig. 1: **Model schematic.** (a) The input weather state(s) are defined on a 0.25° latitude-longitude grid comprising a total of $721 \times 1440 = 1,038,240$ points. Yellow layers in the closeup pop-out window represent the 5 surface variables, and blue layers represent the 6 atmospheric variables that are repeated at 37 pressure levels ($5 + 6 \times 37 = 227$ variables per point in total), resulting in a state representation of $235,680,480$ values. (b) GraphCast predicts the next state of the weather on the grid. (c) A forecast is made by iteratively applying GraphCast to each previous predicted state, to produce a sequence of states which represent the weather at successive lead times. (d) The Encoder component of the GraphCast architecture maps local regions of the input (green boxes) into nodes of the multi-mesh graph representation (green, upward arrows which terminate in the green-blue node). (e) The Processor component updates each multi-mesh node using learned message-passing (heavy blue arrows that terminate at a node). (f) The Decoder component maps the processed multi-mesh features (purple nodes) back onto the grid representation (red, downward arrows which terminate at a red box). (g) The multi-mesh is derived from icosahedral meshes of increasing resolution, from the base mesh (M^0, 12 nodes) to the finest resolution (M^6, $40,962$ nodes), which has uniform resolution across the globe. It contains the set of nodes from M^6, and all the edges from M^0 to M^6. The learned message-passing over the different meshes' edges happens simultaneously, so that each node is updated by all of its incoming edges.

GraphCast

Here we introduce an MLWP approach for global medium-range weather forecasting called "GraphCast", which produces an accurate 10-day forecast in under a minute on a single Google Cloud TPU v4 device, and supports applications including predicting tropical cyclone tracks, atmospheric rivers, and extreme temperatures.

GraphCast takes as input the two most recent states of Earth's weather—the current time and six hours earlier—and predicts the next state of the weather six hours ahead. A single weather state is represented by a 0.25° latitude/longitude grid (721×1440), which corresponds to roughly 28×28 kilometer resolution at the equator (Fig. 1a), where each grid point represents a set of surface and atmospheric variables (listed in Table 1). Like traditional NWP systems, GraphCast is autoregressive: it can be "rolled out" by feeding its own predictions back in as input, to generate an arbitrarily long trajectory of weather states (Fig. 1b–c).

GraphCast is implemented as a neural network architecture, based on GNNs in an "encode-process-decode" configuration (*13, 17*), with a total of 36.7 million parameters (code, weights and demos can be found at `https://github.com/deepmind/graphcast`). Previous GNN-based learned simulators (*18–20*) have been very effective at learning the complex dynamics of fluid and other systems modeled by partial differential equations, which supports their suitability for modeling weather dynamics.

The encoder (Fig. 1d) uses a single GNN layer to map variables (normalized to zero-mean unit-variance) represented as node attributes on the input grid to learned node attributes on an internal "multi-mesh" representation.

The multi-mesh (Fig. 1g) is a graph which is spatially homogeneous, with high spatial resolution over the globe. It is defined by refining a regular icosahedron (12 nodes, 20 faces, 30

edges) iteratively six times, where each refinement divides each triangle into four smaller ones (leading to four times more faces and edges), and reprojecting the nodes onto the sphere. The multi-mesh contains the 40,962 nodes from the highest resolution mesh (which is roughly 1/25 the number of latitude/longitude grid points at 0.25°), and the union of all the edges created in the intermediate graphs, forming a flat hierarchy of edges with varying lengths.

The processor (Fig. 1e) uses 16 unshared GNN layers to perform learned message-passing on the multi-mesh, enabling efficient local and long-range information propagation with few message-passing steps.

The decoder (Fig. 1f) maps the final processor layer's learned features from the multi-mesh representation back to the latitude-longitude grid. It uses a single GNN layer, and predicts the output as a residual update to the most recent input state (with output normalization to achieve unit-variance on the target residual). See Supplementary Materials Section 3 for further architectural details.

During model development, we used 39 years (1979–2017) of historical data from ECMWF's ERA5 (*21*) reanalysis archive. As a training objective, we averaged the mean squared error (MSE) between GraphCast's predicted states over N autoregressive steps and the corresponding ERA5 states, with the error weighted by vertical level (see Supplementary Materials Equation (19)). The value of N was increased incrementally from 1 to 12 (i.e., six hours to three days) over the course of training and the gradient of the loss was computed by backpropagation-through-time (*22*). GraphCast was trained to minimize the training objective using gradient descent which took roughly four weeks on 32 Cloud TPU v4 devices using batch parallelism. See Supplementary Materials Section 4 for further training details.

Consistent with real deployment scenarios, where future information is not available for model development, we evaluated GraphCast on the held out data from the years 2018 onward (see Supplementary Materials Section 5.1).

Verification methods

We verify GraphCast's forecast skill comprehensively by comparing its accuracy to HRES's on a large number of variables, levels, and lead times. We quantify the respective skills of GraphCast, HRES, and ML baselines with two skill metrics: the root mean square error (RMSE) and the anomaly correlation coefficient (ACC).

Of the 227 variable and level combinations predicted by GraphCast at each grid point, we evaluated its skill versus HRES on 69 of them, corresponding to the 13 levels of WeatherBench (*8*) and variables [1] from the ECMWF Scorecard (*24*); see boldface variables and levels in Table 1 and Supplementary Materials Section 1.2 for which HRES cycle was operational during the evaluation period. In addition to the aggregate performance reported in the main text, Supplementary Materials Section 7 provides further detailed evaluations, including other variables, precipitation, regional performance, latitude and pressure level effects, spectral properties, blurring, biases, comparisons to other ML-based forecasts, and effects of model design choices.

In making these comparisons, two key choices underlie how skill is established: (1) the selection of the ground truth for comparison, and (2) a careful accounting of the data assimilation windows used to infer this data from observations. We use ERA5 as the ground truth for evaluating GraphCast, since it was trained to take ERA5 data as input and predict ERA5 data as outputs. However, evaluating HRES forecasts against ERA5 would result in non-zero error on the initial forecast step. Instead, we constructed an "HRES forecast at step 0" (HRES-fc0)

[1] Because precipitation in ERA5 has known biases (*23*), no development decision for GraphCast was made to improve performance on precipitation and GraphCast simply uses precipitation as an auxiliary input/output. Note that precipitation is sparse and non-Gaussian and would have possibly required different modeling decisions than the other variables. Additionally, precipitation is not available in the HRES analysis products in a form amenable to our evaluation protocol (see next paragraphs). Thus, any claim about precipitation prediction is left out of the scope of this work, and we show precipitation evaluation using a different protocol in Supplementary Materials Section 7.1.4 for completeness only.

dataset to use as ground truth for HRES. HRES-fc0 contains the inputs to HRES forecasts at future initializations (see Supplementary Materials Section 1.2), ensuring that each data point is grounded by recent observations and that the zeroth step of HRES forecasts will have zero error.

For a fair comparison, we must ensure that the ERA5 initial conditions for GraphCast were derived from assimilation windows which look no further into the future than those used by HRES. HRES initializations (00z/06z/12z/18z[2]) always assimilate observations 3h into the future while ERA5 initializations assimilate observations 9h into the future at 00z/12z and 3h into the future at 06z/18z. This constrained the choice of initialization times for GraphCast to 06z/18z in all our results. We use the same initializations for HRES when comparing performance up to 3.75 days. Beyond that, HRES archived forecasts are only available from 00z/12z initializations. The transition from 06z/18z to 00z/12z initializations for HRES induces a small discontinuity in our plots that is indicated by a vertically dashed line at the appropriate lead time. Supplementary Materials Section 5 contains further verification details, including details of the comparisons protocol between GraphCast and HRES (Supplementary Materials Section 5.2), and the effect of initialization lookahead on both models' performance (Supplementary Materials Section 5.2.2).

Forecast verification results

We find that GraphCast has greater weather forecasting skill than HRES when evaluated on 10-day forecasts at a horizontal resolution of 0.25° for latitude/longitude and at 13 vertical levels.

Figure 2a–c show how GraphCast (blue lines) outperforms HRES (black lines) on the z500 (geopotential at 500 hPa) "headline" field in terms of RMSE skill, RMSE skill score (i.e., the normalized RMSE difference between model A and baseline B defined as $(\mathrm{RMSE}_A -$

[2]Time in Zulu convention, where 00z means `00:00` UTC

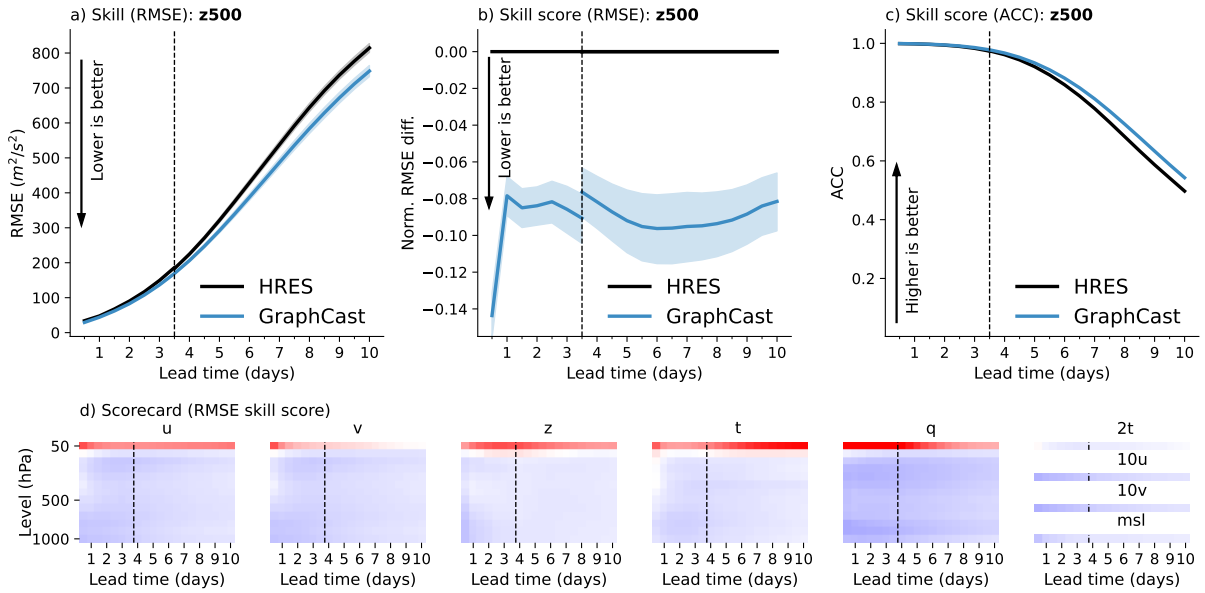

Fig. 2: **Global skill and skill scores for GraphCast and HRES in 2018**. (a) RMSE skill (y-axis) for GraphCast (blue lines) and HRES (black lines), on Z500, as a function of lead time (x-axis). Error bars represent 95% confidence intervals. The vertical dashed line represents 3.5 days, which is the last 12-hour increment of the HRES 06z/18z forecasts. The black line represents HRES, where lead times earlier and later than 3.5 days are from the 06z/18z and 00z/12z initializations, respectively. (b) RMSE skill score (y-axis) for GraphCast versus HRES, on Z500, as a function of lead time (x-axis). Error bars represent 95% confidence intervals for the skill score. We observe a discontinuity in GraphCast's curve because skill scores up to 3.5 days are computed between GraphCast (initialized at 06z/18z) and HRES's 06z/18z initialization, while after 3.5 days skill scores are computed with respect to HRES's 00z/12z initializations. (c) ACC skill (y-axis) for GraphCast (blue lines) and HRES (black lines), on Z500, as a function of lead time (x-axis). (d) Scorecard of RMSE skill scores for GraphCast, with respect to HRES. Each subplot corresponds to one variable: U, V, Z, T, Q, 2T, 10U, 10V, MSL, respectively. The rows of each heatmap correspond to the 13 pressure levels (for the atmospheric variables), from 50 hPa at the top to 1000 hPa at the bottom. The columns of each heatmap correspond to the 20 lead times at 12-hour intervals, from 12 hours on the left to 10 days on the right. Each cell's color represents the skill score, as shown in (b), where blue represents negative values (GraphCast has better skill) and red represents positive values (HRES has better skill).

$\mathrm{RMSE}_B)/\mathrm{RMSE}_B$), and ACC skill. Using z500, which encodes the synoptic-scale pressure distribution, is common in the literature, as it has strong meteorological importance (*8*). The plots show GraphCast has better skill scores across all lead times, with a skill score improvement around 7%–14%. Plots for additional headline variables are in Supplementary Materials Section 7.1.

Figure 2d summarizes the RMSE skill scores for all 1380 evaluated variables and pressure levels, across the 10-day forecasts, in a format analogous to the ECMWF Scorecard. The cell colors are proportional to the skill score, where blue indicates GraphCast had better skill and red indicates HRES had higher skill. GraphCast outperformed HRES on 90.3% of the 1380 targets, and significantly ($p \leq 0.05$, nominal sample size $n \in \{729, 730\}$) outperformed HRES on 89.9% of targets. See Supplementary Materials Section 5.4 for methodology and Supplementary Materials table S4 for p-values, test statistics and effective sample sizes.

The regions of the atmosphere in which HRES had better performance than GraphCast (top rows in red in the scorecards), were disproportionately localized in the stratosphere, and had the lowest training loss weight (see Supplementary Materials Section 7.2.2). When excluding the 50 hPa level, GraphCast significantly outperforms HRES on 96.9% of the remaining 1280 targets. When excluding levels 50 and 100 hPa, GraphCast significantly outperforms HRES on 99.7% of the 1180 remaining targets. When conducting per region evaluations, we found the previous results to generally hold across the globe, as detailed in Supplementary Materials figs. S14 - S16.

We found that increasing the number of autoregressive steps in the MSE loss improves GraphCast performance at longer lead time (see Supplementary Materials Section 7.3.2). It also encourages GraphCast to blur to a degree at longer lead times (see Supplementary Materials fig. S38), which means its forecasts will lie somewhere in between a traditional deterministic forecast, and an ensemble mean. HRES's underlying physical equations, however, do not lead

to blurred predictions. To assess whether GraphCast's relative advantage over HRES on RMSE skill is due to blurrier forecasts better optimizing RMSE, we artificially blurred HRES's forecasts with blurring filters. We fit filters for GraphCast and HRES by minimizing the RMSE between filtered predictions and the models' respective ground truths. We found that RMSE-optimized blurring applied to GraphCast has greater skill than analogous blurring applied to HRES on 88.0% of our 1380 verification targets, which is generally consistent with our above conclusions (see Supplementary Materials Section 7.4). Still, blurrier forecasts may not be desirable for some applications, which we discuss further in the Conclusion.

We also compared GraphCast's performance to the top competing ML-based weather model, Pangu-Weather (*16*), and found GraphCast outperformed it on 99.2% of the 252 targets they presented (see Supplementary Materials Section 6 for details).

Severe event forecasting results

Beyond evaluating GraphCast's forecast skill against HRES's on a wide range of variables and lead times, we also evaluate how its forecasts support predicting severe events, including tropical cyclones tracks, atmospheric rivers, and extreme temperature. These are key downstream applications for which GraphCast is not specifically trained, but which are very important for human activity.

Tropical cyclone tracks

Improving the accuracy of tropical cyclone tracking can help avoid injury and loss of life, as well as reducing economic harm (*25*). A cyclone's existence and trajectory is predicted by applying a tracking algorithm to forecasts of geopotential (Z), horizontal wind (10U/10V, U/V), and mean sea-level pressure (MSL). We implemented a tracking algorithm based on ECMWF's published protocols (*26*) and applied it to GraphCast's forecasts, to produce cyclone track predictions (see

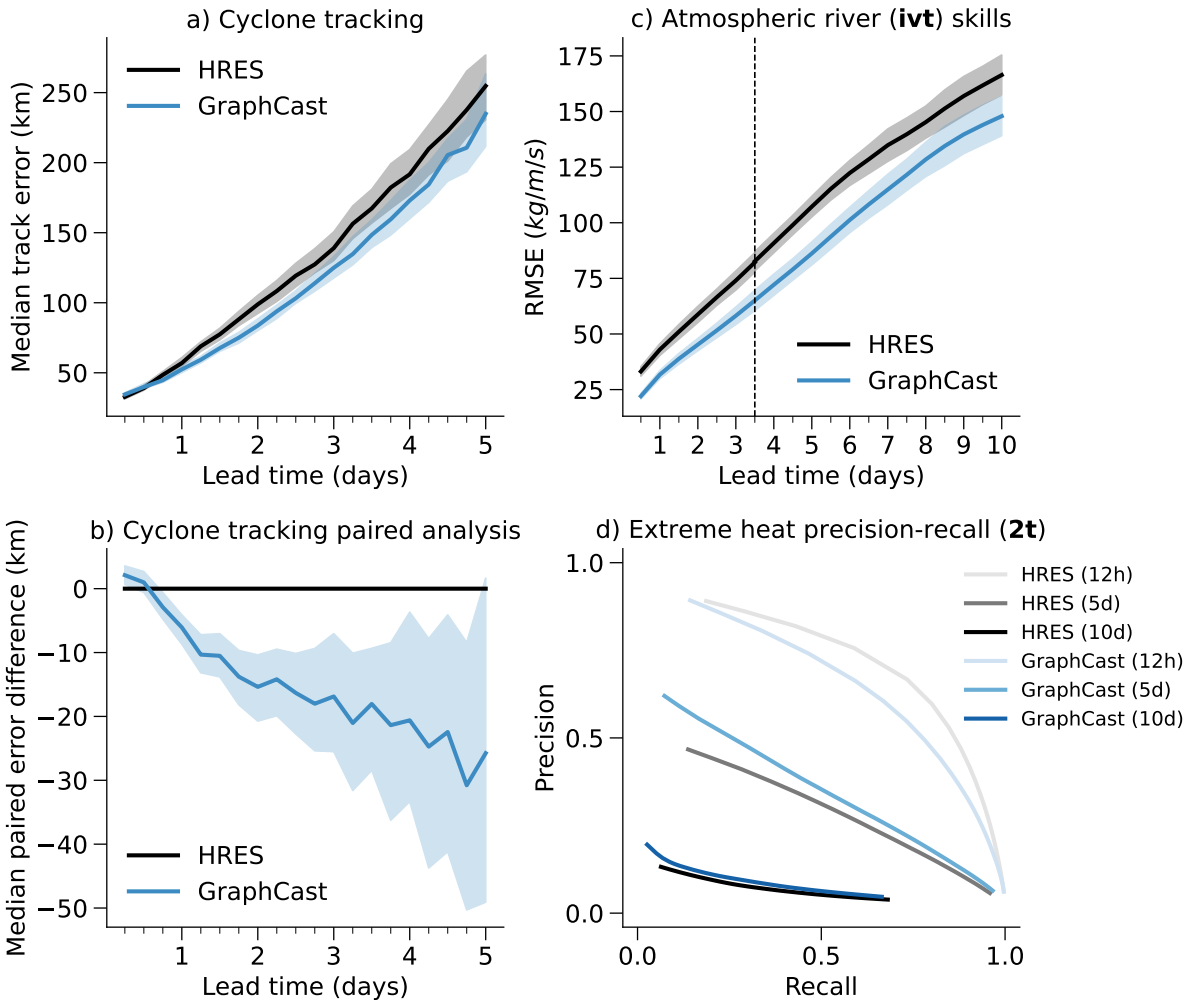

Fig. 3: **Severe-event prediction.** (a) Cyclone tracking performances for GraphCast and HRES. The x-axis represents lead times (in days), and the y-axis represents median track error (in km). Error bars represent bootstrapped 95% confidence intervals for the median. (b) Cyclone tracking paired error difference between GraphCast and HRES. The x-axis represents lead times (in days), and the y-axis represents median paired error difference (in km). Error bars represent bootstrapped 95% confidence intervals for the median difference (see Supplementary Materials Section 8.1). (c) Atmospheric river prediction (IVT) skills for GraphCast and HRES. The x-axis represents lead times (in days), and the y-axis represents RMSE. Error bars are 95% confidence intervals. (d) Extreme heat prediction precision-recall for GraphCast and HRES. The x-axis represents recall, and the y-axis represents precision. The curves represent different precision-recall trade-offs when sweeping over gain applied to forecast signals (see Supplementary Materials Section 8.3).

Supplementary Materials Section 8.1). As a baseline for comparison, we used the operational tracks obtained from HRES's 0.1° forecasts with ECMWF's own tracker, stored in the TIGGE archive (*27, 28*). Each model using the tracker leading to its best performance, we measured errors for both models against the tracks from IBTrACS (*29, 30*), a separate reanalysis dataset of cyclone tracks aggregated from various analysis and observational sources. Consistent with established evaluation of tropical cyclone prediction (*26*), we evaluate all tracks when both GraphCast and HRES detect a cyclone, ensuring that both models are evaluated on the same events, and verify that each model's true-positive rates are similar.

Figure 3a shows GraphCast has lower median track error than HRES over 2018–2021 (median was chosen to resist outliers). As per-track errors for HRES and GraphCast are correlated, we also measured the per-track paired error difference between the two models and found that GraphCast is significantly better than HRES for lead time 18 hours to 4.75 days, as shown in Fig. 3b. The error bars show the bootstrapped 95% confidence intervals for the median (see Supplementary Materials Section 8.1 for details).

Atmospheric rivers

Atmospheric rivers are narrow regions of the atmosphere which are responsible for the majority of the poleward water vapor transport across the mid-latitudes and generate 30%-65% of annual precipitation on the U.S. West Coast (*31*). Their strength can be characterized by the vertically integrated water vapor transport IVT (*32, 33*), indicating whether an event will provide beneficial precipitation or be associated with catastrophic damage (*34*). IVT can be computed from the non-linear combination of the horizontal wind speed (U and V) and specific humidity (Q), which GraphCast predicts. We evaluate GraphCast forecasts over coastal North America and the Eastern Pacific during cold months (Oct–Apr), when atmospheric rivers are most frequent. Despite not being specifically trained to characterize atmospheric rivers, Fig. 3c shows that

GraphCast improves the prediction of IVT compared to HRES, from 25% at short lead time, to 10% at longer horizons (see Supplementary Materials Section 8.2 for details).

Extreme heat and cold

Extreme heat and cold are characterized by large anomalies with respect to typical climatology (*3, 35, 36*), which can be dangerous and disrupt human activities. We evaluate the skill of HRES and GraphCast in predicting events above the top 2% climatology across location, time of day, and month of the year, for 2T at 12-hour, 5-day, and 10-day lead times, for land regions across northern and southern hemisphere over their respective summer months. We plot precision-recall curves (*37*) to reflect different possible trade-offs between reducing false positives (high precision) and reducing false negatives (high recall). For each forecast, we obtain the curve by varying a "gain" parameter that scales the 2T forecast's deviations with respect to the median climatology.

Figure 3d shows GraphCast's precision-recall curves are above HRES's for 5- and 10-day lead times, suggesting GraphCast's forecasts are generally superior than HRES at extreme classification over longer horizons. By contrast, HRES has better precision-recall at the 12-hour lead time, which is consistent with the 2T skill score of GraphCast over HRES being near zero, as shown in Fig. 2d. We generally find these results to be consistent across other variables relevant to extreme heat, such as T850 and Z500 (*36*), other extreme thresholds (5%, 2% and 0.5%), and extreme cold forecasting in winter. See Supplementary Materials Section 8.3 for details.

Effect of training data recency

GraphCast can be re-trained periodically with recent data, which in principle allows it to capture weather patterns that change over time, such as the effects of climate change, and long climate

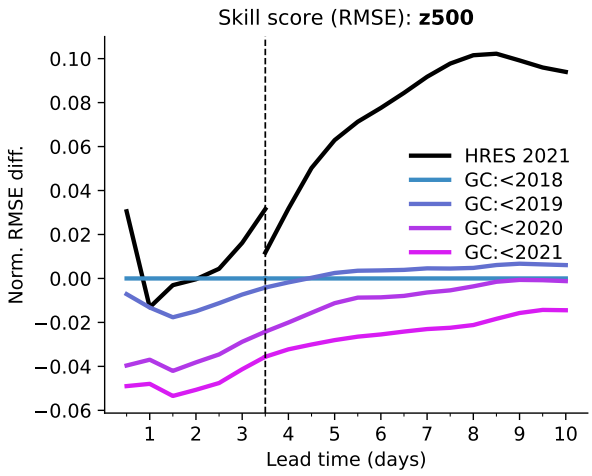

Fig. 4: **Training GraphCast on more recent data.** Each colored line represents GraphCast trained with data ending before a different year, from 2018 (blue) to 2021 (purple). The y-axis represents RMSE skill scores on 2021 test data, for z500, with respect to GraphCast trained up to before 2018, over lead times (x-axis). The vertical dashed line represents 3.5 days, where the HRES 06z/18z forecasts end. The black line represents HRES, where lead times earlier and later than 3.5 days are from the 06z/18z and 00z/12z initializations, respectively.

oscillations. We trained four variants of GraphCast, from scratch, with data that always began in 1979, but ended in 2017, 2018, 2019, and 2020, respectively (we label the variant ending in 2017 as "GraphCast:<2018", etc). We compared their performances to HRES on 2021 test data.

Figure 4 shows the skill scores (normalized by GraphCast:<2018) of the four variants and HRES, for z500. We found that while GraphCast's performance when trained up to before 2018 is still competitive with HRES in 2021, training it up to before 2021 further improves its skill scores (see Supplementary Materials Section 7.1.3). We speculate this recency effect allows recent weather trends to be captured to improve accuracy. This shows that GraphCast's performance can be improved by re-training on more recent data.

Conclusions

GraphCast's forecast skill and efficiency compared to HRES shows MLWP methods are now competitive with traditional weather forecasting methods. Additionally, GraphCast's performance on severe event forecasting, which it was not directly trained for, demonstrates its robustness and potential for downstream value. We believe this marks a turning point in weather forecasting, which helps open new avenues to strengthen the breadth of weather-dependent decision-making by individuals and industries, by making cheap prediction more accurate, more accessible, and suitable for specific applications.

With 36.7 million parameters, GraphCast is a relatively small model by modern ML standards, chosen to keep the memory footprint tractable. And while HRES is released on 0.1° resolution, 137 levels, and up to 1 hour time steps, GraphCast operated on 0.25° latitude-longitude resolution, 37 vertical levels, and 6 hour time steps, because of the ERA5 training data's native 0.25° resolution, and engineering challenges in fitting higher resolution data on hardware. Generally GraphCast should be viewed as a family of models, with the current version being the largest we can practically fit under current engineering constraints, but which have potential to scale much further in the future with greater compute resources and higher resolution data.

One key limitation of our approach is in how uncertainty is handled. We focused on deterministic forecasts and compared against HRES, but the other pillar of ECMWF's IFS, the ensemble forecasting system, ENS, is especially important for quantifying the probability of extreme events and as the skill of the forecast decreases at longer lead times. The non-linearity of weather dynamics means there is increasing uncertainty at longer lead times, which is not well-captured by a single deterministic forecast. ENS addresses this by generating multiple, stochastic forecasts, which approximate a predictive distribution over future weather, however generating multiple forecasts is expensive. By contrast, GraphCast's MSE training objective

encourages it to spatially blur its predictions in the presence of uncertainty, which may not be desirable for some applications where knowing tail, or joint, probabilities of events is important. Building probabilistic forecasts that model uncertainty more explicitly, along the lines of ensemble forecasts, is a crucial next step.

It is important to emphasize that data-driven MLWP relies critically on large quantities of data and their quality, which in the case of models trained on reanalysis, depends on the fidelity of NWPs. Therefore, rich high-quality data sources like ECMWF's MARS archive (*38*) are invaluable. Our approach should not be regarded as a replacement for traditional weather forecasting methods, which have been developed for decades, rigorously tested in many real-world contexts, and offer many features we have not yet explored. Rather our work should be interpreted as evidence that MLWP is able to meet the challenges of real-world forecasting problems and has potential to complement and improve the current best methods.

Beyond weather forecasting, GraphCast can open new directions for other important geospatiotemporal forecasting problems, including climate and ecology, energy, agriculture, and human and biological activity, as well as other complex dynamical systems. We believe that learned simulators, trained on rich, real-world data, will be crucial in advancing the role of machine learning in the physical sciences.

References

1. S. G. Benjamin, J. M. Brown, G. Brunet, P. Lynch, K. Saito, T. W. Schlatter, 100 years of progress in forecasting and NWP applications, *Meteorological Monographs* **59**, 13–1 (2019).

2. P. Bauer, A. Thorpe, G. Brunet, The quiet revolution of numerical weather prediction., *Nature* **525** (2015).

3. I. Lopez-Gomez, A. McGovern, S. Agrawal, J. Hickey, Global extreme heat forecasting using neural weather models, *Artificial Intelligence for the Earth Systems* pp. 1–41 (2022).

4. X. Shi, Z. Gao, L. Lausen, H. Wang, D.-Y. Yeung, W.-k. Wong, W.-c. Woo, Deep learning for precipitation nowcasting: A benchmark and a new model, *Advances in neural information processing systems* **30** (2017).

5. C. K. Sønderby, L. Espeholt, J. Heek, M. Dehghani, A. Oliver, T. Salimans, S. Agrawal, J. Hickey, N. Kalchbrenner, Metnet: A neural weather model for precipitation forecasting, *arXiv preprint arXiv:2003.12140* (2020).

6. S. Ravuri, K. Lenc, M. Willson, D. Kangin, R. Lam, P. Mirowski, M. Fitzsimons, M. Athanassiadou, S. Kashem, S. Madge, *et al.*, Skilful precipitation nowcasting using deep generative models of radar, *Nature* **597**, 672–677 (2021).

7. L. Espeholt, S. Agrawal, C. Sønderby, M. Kumar, J. Heek, C. Bromberg, C. Gazen, R. Carver, M. Andrychowicz, J. Hickey, *et al.*, Deep learning for twelve hour precipitation forecasts, *Nature communications* **13**, 1–10 (2022).

8. S. Rasp, P. D. Dueben, S. Scher, J. A. Weyn, S. Mouatadid, N. Thuerey, WeatherBench: a benchmark data set for data-driven weather forecasting, *Journal of Advances in Modeling Earth Systems* **12**, e2020MS002203 (2020).

9. J. A. Weyn, D. R. Durran, R. Caruana, Can machines learn to predict weather? Using deep learning to predict gridded 500-hPa geopotential height from historical weather data, *Journal of Advances in Modeling Earth Systems* **11**, 2680–2693 (2019).

10. J. A. Weyn, D. R. Durran, R. Caruana, Improving data-driven global weather prediction using deep convolutional neural networks on a cubed sphere, *Journal of Advances in Modeling Earth Systems* **12**, e2020MS002109 (2020).

11. S. Rasp, N. Thuerey, Data-driven medium-range weather prediction with a resnet pretrained on climate simulations: A new model for weatherbench, *Journal of Advances in Modeling Earth Systems* **13**, e2020MS002405 (2021).

12. T. Nguyen, J. Brandstetter, A. Kapoor, J. K. Gupta, A. Grover, ClimaX: A foundation model for weather and climate, *arXiv preprint arXiv:2301.10343* (2023).

13. R. Keisler, Forecasting global weather with graph neural networks, *arXiv preprint arXiv:2202.07575* (2022).

14. J. Pathak, S. Subramanian, P. Harrington, S. Raja, A. Chattopadhyay, M. Mardani, T. Kurth, D. Hall, Z. Li, K. Azizzadenesheli, *et al.*, Fourcastnet: A global data-driven high-resolution weather model using adaptive fourier neural operators, *arXiv preprint arXiv:2202.11214* (2022).

15. T. Kurth, S. Subramanian, P. Harrington, J. Pathak, M. Mardani, D. Hall, A. Miele, K. Kashinath, A. Anandkumar, FourCastNet: Accelerating global high-resolution weather forecasting using adaptive fourier neural operators, *arXiv preprint arXiv:2208.05419* (2022).

16. K. Bi, L. Xie, H. Zhang, X. Chen, X. Gu, Q. Tian, Pangu-Weather: A 3D high-resolution model for fast and accurate global weather forecast, *arXiv preprint arXiv:2211.02556* (2022).

17. P. W. Battaglia, J. B. Hamrick, V. Bapst, A. Sanchez-Gonzalez, V. Zambaldi, M. Malinowski, A. Tacchetti, D. Raposo, A. Santoro, R. Faulkner, *et al.*, Relational inductive biases, deep learning, and graph networks, *arXiv preprint arXiv:1806.01261* (2018).

18. A. Sanchez-Gonzalez, J. Godwin, T. Pfaff, R. Ying, J. Leskovec, P. Battaglia, Learning to simulate complex physics with graph networks, *International Conference on Machine Learning* (PMLR, 2020), pp. 8459–8468.

19. T. Pfaff, M. Fortunato, A. Sanchez-Gonzalez, P. Battaglia, Learning mesh-based simulation with graph networks, *International Conference on Learning Representations* (2021).

20. F. Alet, A. K. Jeewajee, M. B. Villalonga, A. Rodriguez, T. Lozano-Perez, L. Kaelbling, Graph element networks: adaptive, structured computation and memory, *International Conference on Machine Learning* (PMLR, 2019), pp. 212–222.

21. H. Hersbach, B. Bell, P. Berrisford, S. Hirahara, A. Horányi, J. Muñoz-Sabater, J. Nicolas, C. Peubey, R. Radu, D. Schepers, *et al.*, The ERA5 global reanalysis, *Quarterly Journal of the Royal Meteorological Society* **146**, 1999–2049 (2020).

22. P. Werbos, Backpropagation through time: what it does and how to do it, *Proceedings of the IEEE* **78**, 1550-1560 (1990).

23. D. A. Lavers, A. Simmons, F. Vamborg, M. J. Rodwell, An evaluation of ERA5 precipitation for climate monitoring, *Quarterly Journal of the Royal Meteorological Society* **148**, 3152–3165 (2022).

24. T. Haiden, M. Janousek, J. Bidlot, R. Buizza, L. Ferranti, F. Prates, F. Vitart, *Evaluation of ECMWF forecasts, including the 2018 upgrade* (European Centre for Medium Range Weather Forecasts Reading, UK, 2018).

25. A. B. Martinez, Forecast accuracy matters for hurricane damage, *Econometrics* **8**, 18 (2020).

26. L. Magnusson, S. Majumdar, R. Emerton, D. Richardson, M. Alonso-Balmaseda, C. Baugh, P. Bechtold, J. Bidlot, A. Bonanni, M. Bonavita, *et al.*, Tropical cyclone activities at ECMWF, *ECMWF Technical Memorandum* (2021).

27. P. Bougeault, Z. Toth, C. Bishop, B. Brown, D. Burridge, D. H. Chen, B. Ebert, M. Fuentes, T. M. Hamill, K. Mylne, *et al.*, The THORPEX interactive grand global ensemble, *Bulletin of the American Meteorological Society* **91**, 1059–1072 (2010).

28. R. Swinbank, M. Kyouda, P. Buchanan, L. Froude, T. M. Hamill, T. D. Hewson, J. H. Keller, M. Matsueda, J. Methven, F. Pappenberger, M. Scheuerer, H. A. Titley, L. Wilson, M. Yamaguchi, The TIGGE project and its achievements, *Bulletin of the American Meteorological Society* **97**, 49 - 67 (2016).

29. K. R. Knapp, M. C. Kruk, D. H. Levinson, H. J. Diamond, C. J. Neumann, The international best track archive for climate stewardship (IBTrACS) unifying tropical cyclone data, *Bulletin of the American Meteorological Society* **91**, 363–376 (2010).

30. K. R. Knapp, H. J. Diamond, J. P. Kossin, M. C. Kruk, C. J. Schreck, *et al.*, International best track archive for climate stewardship (IBTrACS) project, version 4, https://doi.org/10.25921/82ty-9e16 (2018).

31. W. Chapman, A. Subramanian, L. Delle Monache, S. Xie, F. Ralph, Improving atmospheric river forecasts with machine learning, *Geophysical Research Letters* **46**, 10627–10635 (2019).

32. P. J. Neiman, F. M. Ralph, G. A. Wick, J. D. Lundquist, M. D. Dettinger, Meteorological characteristics and overland precipitation impacts of atmospheric rivers affecting the West Coast of North America based on eight years of ssm/i satellite observations, *Journal of Hydrometeorology* **9**, 22–47 (2008).

33. B. J. Moore, P. J. Neiman, F. M. Ralph, F. E. Barthold, Physical processes associated with heavy flooding rainfall in Nashville, Tennessee, and vicinity during 1–2 May 2010: The role of an atmospheric river and mesoscale convective systems, *Monthly Weather Review* **140**, 358–378 (2012).

34. T. W. Corringham, F. M. Ralph, A. Gershunov, D. R. Cayan, C. A. Talbot, Atmospheric rivers drive flood damages in the western United States, *Science advances* **5**, eaax4631 (2019).

35. L. Magnusson, T. Haiden, D. Richardson, *Verification of extreme weather events: Discrete predictands* (European Centre for Medium-Range Weather Forecasts, 2014).

36. L. Magnusson, Ecmwf confluence wiki, https://confluence.ecmwf.int/display/FCST/202208+-+Heatwave+-+UK (2022).

37. T. Saito, M. Rehmsmeier, The precision-recall plot is more informative than the ROC plot when evaluating binary classifiers on imbalanced datasets, *PloS one* **10**, e0118432 (2015).

38. C. Maass, E. Cuartero, MARS user documentation, https://confluence.ecmwf.int/display/UDOC/MARS+user+documentation (2022).

39. T. Ewalds, Source code for GraphCast, google-deepmind/graphcast: Version 0.1, zenodo, https://doi.org/10.5281/zenodo.10058758 (2023).

40. S. Malardel, N. Wedi, W. Deconinck, M. Diamantakis, C. Kuehnlein, G. Mozdzynski, M. Hamrud, P. Smolarkiewicz, A new grid for the IFS, https://www.ecmwf.int/node/17262 (2016).

41. D. H. Levinson, H. J. Diamond, K. R. Knapp, M. C. Kruk, E. J. Gibney, Toward a homogenous global tropical cyclone best-track dataset, *Bulletin of the American Meteorological Society* **91**, 377–380 (2010).

42. M. C. Kruk, K. R. Knapp, D. H. Levinson, A technique for combining global tropical cyclone best track data, *Journal of Atmospheric and Oceanic Technology* **27**, 680–692 (2010).

43. A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, I. Polosukhin, Attention is all you need, *Advances in neural information processing systems* **30** (2017).

44. P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Lio, Y. Bengio, Graph attention networks, *arXiv preprint arXiv:1710.10903* (2017).

45. M. Fortunato, T. Pfaff, P. Wirnsberger, A. Pritzel, P. Battaglia, Multiscale meshgraphnets, *arXiv preprint arXiv:2210.00612* (2022).

46. P. Battaglia, R. Pascanu, M. Lai, D. Jimenez Rezende, *et al.*, Interaction networks for learning about objects, relations and physics, *Advances in neural information processing systems* **29** (2016).

47. K. R. Allen, Y. Rubanova, T. Lopez-Guevara, W. Whitney, A. Sanchez-Gonzalez, P. Battaglia, T. Pfaff, Learning rigid dynamics with face interaction graph networks, *arXiv preprint arXiv:2212.03574* (2022).

48. P. Ramachandran, B. Zoph, Q. V. Le, Searching for activation functions, *arXiv preprint arXiv:1710.05941* (2017).

49. J. L. Ba, J. R. Kiros, G. E. Hinton, Layer normalization, *arXiv* (2016).

50. B. Bell, H. Hersbach, A. Simmons, P. Berrisford, P. Dahlgren, A. Horányi, J. Muñoz-Sabater, J. Nicolas, R. Radu, D. Schepers, C. Soci, S. Villaume, J.-R. Bidlot, L. Haimberger, J. Woollen, C. Buontempo, J.-N. Thépaut, The era5 global reanalysis: Preliminary extension to 1950, *Quarterly Journal of the Royal Meteorological Society* **147**, 4186-4227 (2021).

51. I. Loshchilov, F. Hutter, Decoupled weight decay regularization, *arXiv preprint arXiv:1711.05101* (2017).

52. D. P. Kingma, J. Ba, Adam: A method for stochastic optimization, *arXiv preprint arXiv:1412.6980* (2014).

53. I. Loshchilov, F. Hutter, SGDR: stochastic gradient descent with restarts, *CoRR* **abs/1608.03983** (2016).

54. T. Chen, B. Xu, C. Zhang, C. Guestrin, Training deep nets with sublinear memory cost, *arXiv preprint arXiv:1604.06174* (2016).

55. J. Bradbury, R. Frostig, P. Hawkins, M. J. Johnson, C. Leary, D. Maclaurin, G. Necula, A. Paszke, J. VanderPlas, S. Wanderman-Milne, Q. Zhang, JAX: composable transformations of Python+NumPy programs, http://github.com/google/jax (2018).

56. T. Hennigan, T. Cai, T. Norman, I. Babuschkin, Haiku: Sonnet for JAX, http://github.com/deepmind/dm-haiku (2020).

57. J. Godwin, T. Keck, P. Battaglia, V. Bapst, T. Kipf, Y. Li, K. Stachenfeld, P. Veličković, A. Sanchez-Gonzalez, Jraph: A library for graph neural networks in JAX., http://github.com/deepmind/jraph (2020).

58. I. Babuschkin, K. Baumli, A. Bell, S. Bhupatiraju, J. Bruce, P. Buchlovsky, D. Budden, T. Cai, A. Clark, I. Danihelka, C. Fantacci, J. Godwin, C. Jones, R. Hemsley, T. Hennigan, M. Hessel, S. Hou, S. Kapturowski, T. Keck, I. Kemaev, M. King, M. Kunesch, L. Martens, H. Merzic, V. Mikulik, T. Norman, J. Quan, G. Papamakarios, R. Ring, F. Ruiz, A. Sanchez, R. Schneider, E. Sezener, S. Spencer, S. Srinivasan, L. Wang, W. Stokowiec, F. Viola, The DeepMind JAX Ecosystem, http://github.com/deepmind (2020).

59. S. Hoyer, J. Hamman, xarray: N-D labeled arrays and datasets in Python, *Journal of Open Research Software* **5** (2017).

60. R. Swinbank, M. Kyouda, P. Buchanan, L. Froude, T. M. Hamill, T. D. Hewson, J. H. Keller, M. Matsueda, J. Methven, F. Pappenberger, *et al.*, The TIGGE project and its achievements, *Bulletin of the American Meteorological Society* **97**, 49–67 (2016).

61. ECMWF, IFS documentation CY45R1 - Part II : Data assimilation, https://www.ecmwf.int/en/elibrary/80893-ifs-documentation-cy45r1-part-ii-data-assimilation (2018).

62. J. R. Driscoll, D. M. Healy, Computing fourier transforms and convolutions on the 2-sphere, *Adv. Appl. Math.* **15**, 202–250 (1994).

63. A. J. Geer, Significance of changes in medium-range forecast scores, *Tellus A: Dynamic Meteorology and Oceanography* **68**, 30229 (2016).

64. T. Haiden, M. Janousek, J.-R. Bidlot, R. Buizza, L. Ferranti, F. Prates, F. Vitart, Evaluation of ECMWF forecasts, including the 2018 upgrade, https://www.ecmwf.int/node/18746 (2018).

65. T. Haiden, M. Janousek, F. Vitart, L. Ferranti, F. Prates, Evaluation of ECMWF forecasts, including the 2019 upgrade, https://www.ecmwf.int/node/19277 (2019).

66. T. Haiden, M. Janousek, F. Vitart, Z. Ben-Bouallegue, L. Ferranti, C. Prates, D. Richardson, Evaluation of ECMWF forecasts, including the 2020 upgrade, https://www.ecmwf.int/node/19879 (2021).

67. T. Haiden, M. Janousek, F. Vitart, Z. Ben-Bouallegue, L. Ferranti, F. Prates, Evaluation of ECMWF forecasts, including the 2021 upgrade, https://www.ecmwf.int/node/20142 (2021).

68. T. Haiden, M. Janousek, F. Vitart, Z. Ben-Bouallegue, L. Ferranti, F. Prates, D. Richardson, Evaluation of ECMWF forecasts, including the 2021 upgrade, https://www.ecmwf.int/node/20469 (2022).

69. M. J. Rodwell, D. S. Richardson, T. D. Hewson, T. Haiden, A new equitable score suitable for verifying precipitation in numerical weather prediction, *Quarterly Journal of the Royal Meteorological Society* **136**, 1344–1363 (2010).

70. T. Haiden, M. J. Rodwell, D. S. Richardson, A. Okagaki, T. Robinson, T. Hewson, Intercomparison of global model precipitation forecast skill in 2010/11 using the seeps score, *Monthly Weather Review* **140**, 2720–2733 (2012).

71. R. North, M. Trueman, M. Mittermaier, M. J. Rodwell, An assessment of the seeps and sedi metrics for the verification of 6 h forecast precipitation accumulations, *Meteorological Applications* **20**, 164–175 (2013).

72. B. D. Santer, R. Sausen, T. M. L. Wigley, J. S. Boyle, K. AchutaRao, C. Doutriaux, J. E. Hansen, G. A. Meehl, E. Roeckner, R. Ruedy, G. Schmidt, K. E. Taylor, Behavior of

tropopause height and atmospheric temperature in models, reanalyses, and observations: Decadal changes, *Journal of Geophysical Research: Atmospheres* **108**, ACL 1-1-ACL 1-22 (2003).

73. ECMWF, IFS documentation CY41R2 - part III: Dynamics and numerical procedures, https://www.ecmwf.int/node/16647 (2016).

74. B. Devaraju, Understanding filtering on the sphere: Experiences from filtering GRACE data, Ph.D. thesis, University of Stuttgart (2015).

75. H. T. Taylor, B. Ward, M. Willis, W. Zaleski, The Saffir-Simpson hurricane wind scale, *Atmospheric Administration: Washington, DC, USA* (2010).

Acknowledgments

In alphabetical order, we thank Kelsey Allen, Charles Blundell, Matt Botvinick, Zied Ben Bouallegue, Michael Brenner, Rob Carver, Matthew Chantry, Marc Deisenroth, Peter Deuben, Marta Garnelo, Ryan Keisler, Dmitrii Kochkov, Christopher Mattern, Piotr Mirowski, Peter Norgaard, Ilan Price, Chongli Qin, Sébastien Racanière, Stephan Rasp, Yulia Rubanova, Kunal Shah, Jamie Smith, Daniel Worrall, and countless others at Alphabet and ECMWF for advice and feedback on our work. We also thank ECMWF for providing invaluable datasets to the research community. The style of the opening paragraph was inspired by D. Fan et al., Science Robotics, 4 (36), (2019).

Funding

All research in this study was funded by Google DeepMind and Alphabet. There was no external funding.

Author contributions

Conceptualization: R.L., A.S., M.W., S.M., P.B. Data curation: RL, A.S., M.W., A.M., P.B. Formal analysis: R.L., A.S., M.W., P.W., M.F., F.A., S.R., T.E., Z.E., W.H., A.P., S.M., P.B. Investigation: R.L., A.S., M.W., P.W., M.F., F.A., S.R., T.E., Z.E., W.H., A.P., S.M., P.B. Methodology: R.L., A.S., M.W., P.W., M.F., F.A., S.R., T.E., Z.E., W.H., A.P., S.M., P.B. Project administration: R.L., A.S., M.W., G.H., O.V., J.S., S.M., P.B. Software: R.L., A.S., M.W., P.W., M.F., F.A., S.R., T.E., Z.E., W.H., A.P., P.B. Supervision: R.L., S.M., P.B. Validation: R.L., A.S., M.W., P.W., M.F., F.A., S.R., T.E., Z.E., W.H., A.P., S.M., P.B. Visualization: R.L., A.S., M.W., F.A., P.B. Writing - original draft: R.L., A.S., M.W., P.W., M.F., F.A., S.R., T.E., Z.E., SH, A.P., S.M., P.B. Writing - review and editing: R.L., A.S., M.W., P.W., M.F., F.A., S.R., T.E., Z.E., SH, A.P., S.M., P.B.

Competing interests

This work was done in the course of employment at Google DeepMind, with no other competing financial interests. A.S., R.L., P.B., M.W., P.W., M.F. and A.P. have filed a provisional patent application relating to machine learning for learned medium-range global weather forecasting (US Provisional App. no. US63/435,163).

Data and Materials Availability

GraphCast's code and trained weights are publicly available on GitHub https://github.com/deepmind/graphcast (*39*). This work used publicly available data from the European Centre for Medium Range Forecasting (ECMWF). We use the ECMWF archive (expired real-time) products for ERA5, HRES and TIGGE products, whose use is governed by the Creative Commons Attribution 4.0 International (CC BY 4.0). We use IBTrACS Version 4 from

https://www.ncei.noaa.gov/products/international-best-track-archive and reference (*29, 30*) as required. The Earth texture in figure 1 is used under CC BY 4.0 from https://www.solarsystemscope.com/textures/.

Supplementary Materials for

Learning skillful medium-range global weather forecasting

Remi Lam,[1*] Alvaro Sanchez-Gonzalez,[1*] Matthew Willson,[1*] Peter Wirnsberger,[1*] Meire Fortunato,[1*] Ferran Alet,[1*] Suman Ravuri,[1*] Timo Ewalds,[1] Zach Eaton-Rosen,[1] Weihua Hu,[1] Alexander Merose,[2] Stephan Hoyer,[2] George Holland,[1] Oriol Vinyals,[1] Jacklynn Stott,[1] Alexander Pritzel,[1] Shakir Mohamed,[1] Peter Battaglia[1]

[1]Google DeepMind, [2]Google Research, [*]Equal contributions.

Corresponding authors: remilam@google.com, alvarosg@google.com, matthjw@google.com, shakir@google.com, peterbattaglia@google.com

This PDF file includes:

- Materials and Methods
- Supplementary Text
- Figs. S1 to S53
- Tables S1 to S4
- References *(40-75)*

Contents

Materials and Methods

1 Datasets

In this section, we give an overview of the data we used to train and evaluate GraphCast (Supplements Section 1.1), the data defining the forecasts of the NWP baseline HRES, as well as HRES-fc0, which we use as ground truth for HRES (Supplements Section 1.2). Finally, we describe the data used in the tropical cyclone tracking analysis (Section 1.3).

We constructed multiple datasets for training and evaluation, comprised of subsets of ECMWF's data archives and IBTrACS (*29, 30*). We generally distinguish between the source data, which we refer to as "archive" or "archived data", versus the datasets we have built from these archives, which we refer to as "datasets".

1.1 ERA5

For training and evaluating GraphCast, we built our datasets from a subset of ECMWF's ERA5 (*21*)[3] archive, which is a large corpus of data that represents the global weather from 1959 to the present, at 0.25° latitude/longitude resolution, and 1 hour increments, for hundreds of static, surface, and atmospheric variables. The ERA5 archive is based on *reanalysis*, which uses ECMWF's HRES model (cycle 42r1) that was operational for most of 2016 (see Table S2), within ECMWF's 4D-Var data assimilation system. ERA5 assimilated 12-hour windows of observations, from 21z-09z and 09z-21z, as well as previous forecasts, into a dense representation of the weather's state, for each historical date and time.

Our ERA5 dataset contains a subset of available variables in ECMWF's ERA5 archive (Table S1), on 37 pressure levels[4]: 1, 2, 3, 5, 7, 10, 20, 30, 50, 70, 100, 125, 150, 175, 200,

[3] See ERA5 documentation: https://confluence.ecmwf.int/display/CKB/ERA5.

[4] We follow common practice of using pressure as our vertical coordinate, instead of altitude. A "pressure level" is a field of altitudes with equal pressure. E.g., "pressure level 500 hPa" corresponds to the field of altitudes for

225, 250, 300, 350, 400, 450, 500, 550, 600, 650, 700, 750, 775, 800, 825, 850, 875, 900, 925, 950, 975, 1000 hPa. The range of years included was 1979-01-01 to 2022-01-10, which were downsampled to 6 hour time intervals (corresponding to 00z, 06z, 12z and 18z each day). The downsampling is performed by subsampling, except for the total precipitation, which is accumulated for the 6 hours leading up to the corresponding downsampled time.

1.2 HRES

Evaluating the HRES model baseline requires two separate sets of data, namely the forecast data and the ground truth data, which are summarized in the subsequent sub-sections. The HRES versions which were operational during our test years are shown in Table S2.

HRES operational forecasts HRES is generally considered to be the most accurate deterministic NWP-based weather model in the world, so to evaluate the HRES baseline, we built a dataset of HRES's archived historical forecasts. HRES is regularly updated by ECMWF, so these forecasts represent the latest HRES model at the time the forecasts were made. The forecasts were downloaded at their native representation (which uses spherical harmonics and an octahedral reduced Gaussian grid, TCo1279 (*40*)), and roughly corresponds to 0.1° latitude/longitude resolution. We then spatially downsampled the forecasts to a 0.25° latitude/longitude grid (to match ERA5's resolution) using ECMWF's Metview library, with default `regrid` parameters. We temporally downsampled them to 6 hour intervals. There are two groups of HRES forecasts: those initialized at 00z/12z which are released for 10 day horizons, and those initialized at 06z/18z which are released for 3.75 day horizons.

which the pressure is 500 hPa. The relationship between pressure and altitude is determined by the geopotential variable.

Type	Variable name	Short name	ECMWF Parameter ID	Role (accumulation period, if applicable)
Atmospheric	Geopotential	z	129	Input/Predicted
Atmospheric	Specific humidity	q	133	Input/Predicted
Atmospheric	Temperature	t	130	Input/Predicted
Atmospheric	U component of wind	u	131	Input/Predicted
Atmospheric	V component of wind	v	132	Input/Predicted
Atmospheric	Vertical velocity	w	135	Input/Predicted
Single	2 metre temperature	2t	167	Input/Predicted
Single	10 metre u wind component	10u	165	Input/Predicted
Single	10 metre v wind component	10v	166	Input/Predicted
Single	Mean sea level pressure	msl	151	Input/Predicted
Single	Total precipitation	tp	228	Input/Predicted (6h)
Single	TOA incident solar radiation	tisr	212	Input (1h)
Static	Geopotential at surface	z	129	Input
Static	Land-sea mask	lsm	172	Input
Static	Latitude	n/a	n/a	Input
Static	Longitude	n/a	n/a	Input
Clock	Local time of day	n/a	n/a	Input
Clock	Elapsed year progress	n/a	n/a	Input

Table S1: **ECMWF variables used in our datasets.** The "Type" column indicates whether the variable represents a *static* property, a time-varying *single*-level property (e.g., surface variables are included), or a time-varying *atmospheric* property. The "Variable name" and "Short name" columns are ECMWF's labels. The "ECMWF Parameter ID" column is a ECMWF's numeric label, and can be used to construct the URL for ECMWF's description of the variable, by appending it as suffix to the following prefix, replacing "ID" with the numeric code: `https://apps.ecmwf.int/codes/grib/param-db/?id=ID`. The "Role" column indicates whether the variable is something our model takes as input and predicts, or only uses as input context (the double horizontal line separates predicted from input-only variables, to make the partitioning more visible).

IFS cycle	Dates of operation	Used in ERA5	HRES evaluation year(s)
42r1	2016-03-08 – 2016-11-21	✓	–
43r1	2016-11-22 – 2017-07-10		–
43r3	2017-07-11 – 2018-06-04		2018
45r1	2018-06-05 – 2019-06-10		2018, 2019
46r1	2019-06-11 – 2020-06-29		2019, 2020
47r1	2020-06-30 – 2021-05-10		2020, 2021
47r2	2021-05-11 – 2021-10-11		2021
47r3	2021-10-12 – present		2021, 2022

Table S2: **0.1° resolution IFS cycles since 2016.** The table shows every IFS cycle that operated at 0.1° latitude/longitude resolution. The columns represent the IFS cycle version, its dates of operation, whether it was used for data assimilation for ERA5, and the years it was used as a baseline for comparing to GraphCast in our results evaluation. See https://www.ecmwf.int/en/forecasts/documentation-and-support/changes-ecmwf-model for the full cycle release schedule.

HRES-fc0 For evaluating the skill of the HRES operational forecasts, we constructed a ground truth dataset, "HRES-fc0", based on ECMWF's HRES operational forecast archive. This dataset comprises the initial time step of each HRES forecast, at initialization times 00z, 06z, 12z, and 18z (see Figure S1). The HRES-fc0 data is similar to the ERA5 data, but it is assimilated using the latest ECMWF NWP model at the forecast time. Assimilation for 00z, 06z, 12z and 18z uses observations up to +3h ahead (i.e., lookahead) of the corresponding date and time (see Supplement Section 5.2.2). Note, ECMWF also provides an archive of "HRES Analysis" data, which is distinct from our HRES-fc0 dataset. The HRES-Analysis dataset and HRES-fc0 sometimes differ due to the presence or absence of land surface analysis, and differences in the data assimilation processes. Therefore we use HRES-fc0 as ground truth to ensure that HRES has zero error at the zeroth timestep, and to minimize error at short lead times due to differences in the input and the ground truth data sources. In the ECMWF Scorecard, all models are evaluated against HRES-Analysis, however errors at short lead times due to these discrepancies would be present across all models, rather than making one look apparently worse, as would be the case if we were to use it for verifying HRES.

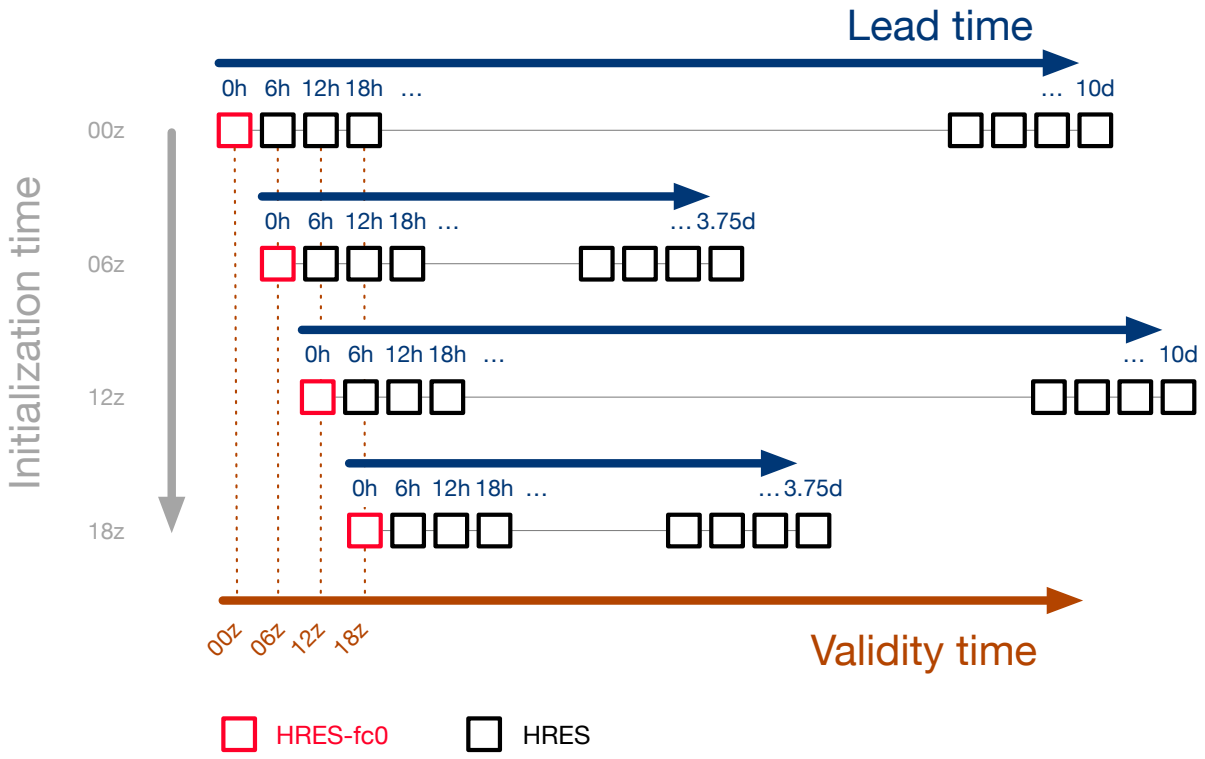

Fig. S1: **Schematic of HRES-fc0.** Each horizontal line represent a forecast made by HRES, initialized at a different time (grey axis). HRES forecasts initialized from 00z and 12z make predictions up to 10 days lead time (blue axis), while HRES forecasts initialized from 06z and 18z make predictions up to 3.75 days. Each square represent a state predicted by HRES, by 6 hours increments (smaller time steps are omitted from the schematic, as well as states in the middle of a forecast trajectory). Red squares represent the forecast at time 0 for each HRES forecast, and defines the data points included in HRES-fc0. The brown axis represents the validity time and allows visualizing the alignment of predictions from different initialization time. For instance, the error of the prediction made by HRES, initialized at 06z (second row of squares from the top), at 12h lead time, i.e., 18z validity time (3rd square from the left) would be measured against the first step of the HRES forecast initialized at 18z (red square from the last row of square).

HRES NaN handling A very small subset of the values from the ECMWF HRES archive for the variable geopotential at 850hPa (z850) and 925hPa (z925) are not numbers (NaN). These NaN's seem to be distributed uniformly across the 2016-2021 range and across forecast times. This represents about 0.00001% of the pixels for z850 (1 pixel every ten 1440 x 721 latitude-longitude frames), 0.00000001% of the pixels for z925 (1 pixel every ten thousand 1440 x 721 latitude-longitude frames) and has no measurable impact on performance. For easier comparison, we filled these rare missing values with the weighted average of the immediate neighboring pixels. We used a weight of 1 for side-to-side neighbors and 0.5 weights for diagonal neighbors[5].

1.3 Tropical cyclone datasets

For our analysis of tropical cyclone tracking, we used the IBTrACS (*29, 30, 41, 42*) archive to construct the ground truth dataset. This includes historical cyclone tracks from around a dozen authoritative sources. Each track is a time series, at 6-hour intervals (00z, 06z, 12z, 18z), where each timestep represents the eye of the cyclone in latitude/longitude coordinates, along with the corresponding Saffir-Simpson category and other relevant meteorological features at that point in time.

For the HRES baseline, we used the TIGGE archive, which provides cyclone tracks estimated with the operational tracker, from HRES's forecasts at 0.1° resolution (*27, 28*). The data is stored as XML files available for download under https://confluence.ecmwf.int/display/TIGGE/Tools. To convert the data into a format suitable for further post-processing and analysis, we implemented a parser that extracts cyclone tracks for the years of interest. The relevant sections (tags) in the XML files are those of type "forecast", which typically contain multiple tracks corresponding to different initial forecast times. Within these tags,

[5]In the extremely rare cases that the neighbors were also NaN's, we dropped both the NaN neighbor and the opposed neighbor from the weighted average.

we then extract the cyclone name (tag “cycloneName”), the latitude (tag “latitude”) and the longitude (tag “longitude”) values, and the valid time (tag “validTime”).

See Section 8.1 for details of the tracker algorithm and results.

2 Notation and problem statement

In this section, we define useful time notations use throughout the paper (Section 2.1), formalize the general forecasting problem we tackle (Section 2.2), and detail how we model the state of the weather (Section 2.3).

2.1 Time notation

The time notation used in forecasting can be confusing, involving a number of different time symbols, e.g., to denote the initial forecast time, validity time, forecast horizon, etc. We therefore introduce some standardized terms and notation for clarity and simplicity. We refer to a particular point in time as "date-time", indicated by calendar date and UTC time. For example, `2018-06-21_18:00:00` means June 21, 2018, at 18:00 UTC. For shorthand, we also sometimes use the Zulu convention, i.e., 00z, 06z, 12z, 18z mean `00:00`, `06:00`, `12:00`, `18:00` UTC, respectively. We further define the following symbols:

- t: Forecast time step index, which indexes the number of steps since the forecast was initialized.
- T: Forecast horizon, which represents the total number of steps in a forecast.
- d: Validity time, which indicates the date-time of a particular weather state.
- d_0: Forecast initialization time, indicating the validity time of a forecast's initial inputs.
- Δd: Forecast step duration, indicating how much time elapses during one forecast step.
- τ: Forecast lead time, which represents the elapsed time in the forecast (i.e., $\tau = t\Delta d$).

2.2 General forecasting problem statement

Let Z^d denote the true state of the global weather at time d. The time evolution of the true weather can be represented by an underlying discrete-time dynamics function, Φ, which generates the state at the next time step (Δd in the future) based on the current one, i.e., $Z^{d+\Delta d} = \Phi(Z^d)$. We then obtain a trajectory of T future weather states by applying Φ autoregressively T times,

$$Z^{d+\Delta d:d+T\Delta d} = (\underbrace{\Phi(Z^d), \Phi(Z^{d+\Delta d}), \ldots, \Phi(Z^{d+(T-1)\Delta d})}_{1 \ldots T \text{ autoregressive iterations}}). \tag{1}$$

Our goal is to find an accurate and efficient model, ϕ, of the true dynamics function, Φ, that can efficiently forecast the state of the weather over some forecast horizon, $T\Delta d$. We assume that we cannot observe Z^d directly, but instead only have some partial observation X^d, which is an incomplete representation of the state information required to predict the weather perfectly. Because X^d is only an approximation of the instantaneous state Z^d, we also provide ϕ with one or more past states, $X^{d-\Delta d}, X^{d-2\Delta d}, ...$, in addition to X^d. The model can then, in principle, leverage this additional context information to approximate Z^d more accurately. Thus ϕ predicts a future weather state as,

$$\hat{X}^{d+\Delta d} = \phi(X^d, X^{d-\Delta d}, ...). \tag{2}$$

Analogous to Equation (1), the prediction $\hat{X}^{d+\Delta d}$ can be fed back into ϕ to autoregressively produce a full forecast,

$$\hat{X}^{d+\Delta d:d+T\Delta d} = (\underbrace{\phi(X^d, X^{d-\Delta d}, ...), \phi(\hat{X}^{d+\Delta d}, X^d, ...), \ldots, \phi(\hat{X}^{d+(T-1)\Delta d}, \hat{X}^{d+(T-2)\Delta d}, ...)}_{1 \ldots T \text{ autoregressive iterations}}). \tag{3}$$

We assess the forecast quality, or skill, of ϕ by quantifying how well the predicted trajectory, $\hat{X}^{d+\Delta d:d+T\Delta d}$, matches the ground-truth trajectory, $X^{d+\Delta d:d+T\Delta d}$. However, it is important

to highlight again that $X^{d+\Delta d:d+T\Delta d}$ only comprises our observations of $Z^{d+\Delta d:d+T\Delta d}$, which itself is unobserved. We measure the consistency between forecasts and ground truth with an objective function,

$$\mathcal{L}\left(\hat{X}^{d+\Delta d:d+T\Delta d}, X^{d+\Delta d:d+T\Delta d}\right),$$

which is described explicitly in Section 5.

In our work, the temporal resolution of data and forecasts was always $\Delta d = 6$ hours with a maximum forecast horizon of 10 days, corresponding to a total of $T = 40$ steps. Because Δd is a constant throughout this paper, we can simplify the notation using $(X^t, X^{t+1}, \ldots, X^{t+T})$ instead of $(X^d, X^{d+\Delta d}, \ldots, X^{d+T\Delta d})$, to index time with an integer instead of a specific date-time.

2.3 Modeling ECMWF weather data

For training and evaluating models, we treat our ERA5 dataset as the ground truth representation of the surface and atmospheric weather state. As described in Section 1.2, we used the HRES-fc0 dataset as ground truth for evaluating the skill of HRES.

In our dataset, an ERA5 weather state X^t comprises all variables in Table S1 (see next paragraph for details of the variables), at a 0.25° horizontal latitude-longitude resolution with a total of $721 \times 1440 = 1,038,240$ grid points and 37 vertical pressure levels. The atmospheric variables are defined at all pressure levels and the set of (horizontal) grid points is given by $G_{0.25^\circ} = \{-90.0, -89.75, \ldots, 90.0\} \times \{-179.75, -179.5, \ldots, 180.0\}$. These variables are uniquely identified by their short name (and the pressure level, for atmospheric variables). For example, the surface variable "2 metre temperature" is denoted `2T`; the atmospheric variable "Geopotential" at pressure level 500 hPa is denoted `Z500`. Note, only the "predicted" variables are output by our model, because the "input"-only variables are forcings that are known apri-

ori, and simply appended to the state on each time-step. We ignore them in the description for simplicity.

The input and predicted variables used in our model are shown in Table 1 and Table S1. Of the predicted variables (indicated in Table S1 with "Input/Predicted" in the "Role" column), 5 are surface variables, and 6 are atmospheric variables at 37 pressure levels, giving a total of 227 ($5 + 6 \times 37$) target variables. Several other static and/or external variables were also provided as input context for our model (indicated in Table S1 with "Input" in the "Role" column). The static/external variables include information such as the geometry of the grid/mesh, orography (surface geopotential), land-sea mask and radiation at the top of the atmosphere.

We refer to the subset of variables in X^t that correspond to a particular grid point i (1,038,240 in total) as $\mathbf{x}_i^t$, and to each variable j of the 227 target variables as $x_{i,j}^t$. The full state representation X^t therefore contains a total of $721 \times 1440 \times (5 + 6 \times 37) = 235,680,480$ values. Note, at the poles, the 1440 longitude points are equal, so the actual number of distinct grid points is slightly smaller.

One thing to note is that the coordinate system we use is not special, and can be changed to suit particular use cases. For example, using RMSE as a training objective to optimize wind vectors in Cartesian coordinates incentivizes reducing the wind magnitude under predictive uncertainty, which may not be desirable. Using polar coordinates (magnitude and angular direction) may be more desirable in this case.

3 GraphCast model

This section provides a detailed description of GraphCast, starting with the autoregressive generation of a forecast (Section 3.1), an overview of the architecture in plain language (Section 3.2), followed by a technical description the all the graphs defining GraphCast (Section 3.3), its encoder (Section 3.4), processor (Section 3.5), and decoder (Section 3.6), as well as all the normalization and parameterization details (Section 3.7).

3.1 Generating a forecast

Our GraphCast model is defined as a one-step learned simulator that takes the role of ϕ in Equation (2) and predicts the next step based on two consecutive input states,

$$\hat{X}^{t+1} = \text{GraphCast}(X^t, X^{t-1}). \tag{4}$$

As in Equation (3), we can apply GraphCast iteratively to produce a forecast

$$\hat{X}^{t+1:t+T} = (\underbrace{\text{GraphCast}(X^t, X^{t-1}), \text{GraphCast}(\hat{X}^{t+1}, X^t), \ldots, \text{GraphCast}(\hat{X}^{t+T-1}, \hat{X}^{t+T-2})}_{1 \ldots T \text{ autoregressive iterations}}) \tag{5}$$

of arbitrary length, T. This is illustrated in Figure 1b,c. We found, in early experiments, that two input states yielded better performance than one, and that three did not help enough to justify the increased memory footprint.

3.2 Architecture overview

GraphCast is implemented using GNNs in an "encode-process-decode" configuration (*17*), as depicted in Figure 1d,e,f, where the encoder maps (surface and atmospheric) features on the input latitude-longitude grid to a multi-mesh, the processor performs many rounds of message-passing on the multi-mesh, and the decoder maps the multi-mesh features back to the output latitude-longitude grid (see Figure 1). GNN-based learned simulators are very effective at

learning complex physical dynamics of fluids and other materials (*18, 19*), as the structure of their representations and computations are analogous to learned finite element solvers (*20*). A key advantage of GNNs is that the input graph's structure determines what parts of the representation interact with one another via learned message-passing, allowing arbitrary patterns of spatial interactions over any range. By contrast, a convolutional neural network (CNN) is restricted to computing interactions within local patches (or, in the case of dilated convolution, over regularly strided longer ranges). And while Transformers (*43*) can also compute arbitrarily long-range computations, they do not scale well with very large inputs (e.g., the 1 million-plus grid points in GraphCast's global inputs) because of the quadratic memory complexity induced by computing all-to-all interactions. Contemporary extensions of Transformers often sparsify possible interactions to reduce the complexity, which in effect makes them analogous to GNNs (e.g., graph attention networks (*44*)).

The way we capitalize on the GNN's ability to model arbitrary sparse interactions is by introducing GraphCast's new internal "multi-mesh" representation, which allows long-range interactions within few message-passing steps and has generally homogeneous spatial resolution over the globe. This is in contrast with a latitude-longitude grid which induce a non-uniform distribution of grid points. Using the latitude-longitude grid is not an advisable representation due to its spatial inhomogeneity, and high resolution at the poles which demands disproportionate compute resources.

Our multi-mesh is constructed by first dividing a regular icosahedron (12 nodes and 20 faces) iteratively 6 times to obtain a hierarchy of icosahedral meshes with a total of 40,962 nodes and 81,920 faces on the highest resolution. We leveraged the fact that the coarse-mesh nodes are subsets of the fine-mesh nodes, which allowed us to superimpose edges from all levels of the mesh hierarchy onto the finest-resolution mesh. This procedure yields a multi-scale set of meshes, with coarse edges bridging long distances at multiple scales, and fine edges capturing

local interactions. Figure 1g shows each individual refined mesh, and Figure 1e shows the full multi-mesh.

GraphCast's encoder (Figure 1d) first maps the input data, from the original latitude-longitude grid, into learned features on the multi-mesh, using a GNN with directed edges from the grid points to the multi-mesh. The processor (Figure 1e) then uses a 16-layer deep GNN to perform learned message-passing on the multi-mesh, allowing efficient propagation of information across space due to the long-range edges. The decoder (Figure 1f) then maps the final multi-mesh representation back to the latitude-longitude grid using a GNN with directed edges, and combines this grid representation, $\hat{Y}^{t+k}$, with the input state, $\hat{X}^{t+k}$, to form the output prediction, $\hat{X}^{t+k+1} = \hat{X}^{t+k} + \hat{Y}^{t+k}$.

The encoder and decoder do not require the raw data to be arranged in a regular rectilinear grid, and can also be applied to arbitrary mesh-like state discretizations (*20*). The general architecture builds on various GNN-based learned simulators which have been successful in many complex fluid systems and other physical domains (*18, 19, 45*). Similar approaches were used in weather forecasting (*13*), with promising results.

On a single Cloud TPU v4 device[6], GraphCast can generate a 0.25° resolution, 10-day forecast (at 6-hour steps) in under 60 seconds. For comparison, ECMWF's IFS system runs on a 11,664-core cluster, and generates a 0.1° resolution, 10-day forecast (released at 1-hour steps for the first 90 hours, 3-hour steps for hours 93-144, and 6-hour steps from 150-240 hours, in about an hour of compute time (*8*). See the HRES release details here: https://www.ecmwf.int/en/forecasts/datasets/set-i.

[6]For information about Cloud TPU v4 performance, memory and energy consumption, see https://cloud.google.com/tpu/docs/system-architecture-tpu-vm#tpu_v4.

3.3 GraphCast's graph

The model operates on a graph $\mathcal{G}(\mathcal{V}^{\mathrm{G}}, \mathcal{V}^{\mathrm{M}}, \mathcal{E}^{\mathrm{M}}, \mathcal{E}^{\mathrm{G2M}}, \mathcal{E}^{\mathrm{M2G}})$, defined in detail in the subsequent paragraphs.

Grid nodes $\mathcal{V}^{\mathrm{G}}$ represents the set containing each of the grid nodes v_i^{G}. Each grid node represents a vertical slice of the atmosphere at a given latitude-longitude point, i. The features associated with each grid node v_i^{G} are $\mathbf{v}_i^{\mathrm{G,features}} = [\mathbf{x}_i^{t-1}, \mathbf{x}_i^t, \mathbf{f}_i^{t-1}, \mathbf{f}_i^t, \mathbf{f}_i^{t+1}, \mathbf{c}_i]$, where $\mathbf{x}_i^t$ is the time-dependent weather state X^t corresponding to grid node v_i^{G} and includes all the predicted data variables for all 37 atmospheric levels as well as surface variables. The forcing terms $\mathbf{f}^t$ consist of time-dependent features that can be computed analytically, and do not need to be predicted by GraphCast. They include the total incident solar radiation at the top of the atmosphere, accumulated over 1 hour, the sine and cosine of the local time of day (normalized to [0, 1)), and the sine and cosine of the of year progress (normalized to [0, 1)). The constants $\mathbf{c}_i$ are static features: the binary land-sea mask, the geopotential at the surface, the sine of the latitude, and the sine and cosine of the longitude. At 0.25° resolution, there is a total of $721 \times 1440 = 1,038,240$ grid nodes, each with (5 *surface variables* + 6 *atmospheric variables* $\times$ 37 *levels*) $\times$ 2 *steps* + 5 *forcings* $\times$ 3 *steps* + 5 *constant* = 474 input features.

Mesh nodes $\mathcal{V}^{\mathrm{M}}$ represents the set containing each of the mesh nodes v_i^{M}. Mesh nodes are placed uniformly around the globe in a R-refined icosahedral mesh M^R. M^0 corresponds to a unit-radius icosahedron (12 nodes and 20 triangular faces) with faces parallel to the poles (see Figure 1g). The mesh is iteratively refined $M^r \rightarrow M^{r+1}$ by splitting each triangular face into 4 smaller faces, resulting in an extra node in the middle of each edge, and re-projecting the new nodes back onto the unit sphere.[7] Features $\mathbf{v}_i^{\mathrm{M,features}}$ associated with each mesh node v_i^{M} are the

[7]Note this split and re-project mechanism leads to a maximum difference of 16.4% and standard deviation of 6.5% in triangle edge lengths across the mesh.

cosine of the latitude, and the sine and cosine of the longitude. GraphCast works with a mesh that has been refined $R = 6$ times, M^6, resulting in 40,962 mesh nodes (see Supplementary Table S3), each with the 3 input features.

Refinement	0	1	2	3	4	5	6
Num Nodes	12	42	162	642	2,562	10,242	40,962
Num Faces	20	80	320	1,280	5,120	20,480	81,920
Num Edges	60	240	960	3,840	15,360	61,440	245,760
Num Multilevel Edges	60	300	1,260	5,100	20,460	81,900	327,660

Table S3: **Multi-mesh statistics.** Statistics of the multilevel refined icosahedral mesh as function of the refinement level R. Edges are considered to be bi-directional and therefore we count each edge in the mesh twice (once for each direction).

Mesh edges $\mathcal{E}^{\mathrm{M}}$ are bidirectional edges added between mesh nodes that are connected in the mesh. Crucially, mesh edges are added to $\mathcal{E}^{\mathrm{M}}$ for all levels of refinement, i.e., for the finest mesh, M^6, as well as for M^5, M^4, M^3, M^2, M^1 and M^0. This is straightforward because of how the refinement process works: the nodes of M^{r-1} are always a subset of the nodes in M^r. Therefore, nodes introduced at lower refinement levels serve as hubs for longer range communication, independent of the maximum level of refinement. The resulting graph that contains the joint set of edges from all of the levels of refinement is what we refer to as the "multi-mesh". See Figure 1e,g for a depiction of all individual meshes in the refinement hierarchy, as well as the full multi-mesh.

For each edge $e^{\mathrm{M}}_{v^{\mathrm{M}}_{\mathrm{s}} \to v^{\mathrm{M}}_{\mathrm{r}}}$ connecting a sender mesh node $v^{\mathrm{M}}_{\mathrm{s}}$ to a receiver mesh node $v^{\mathrm{M}}_{\mathrm{r}}$, we build edge features $\mathbf{e}^{\mathrm{M,features}}_{v^{\mathrm{M}}_{\mathrm{s}} \to v^{\mathrm{M}}_{\mathrm{r}}}$ using the position on the unit sphere of the mesh nodes. This includes the length of the edge, and the vector difference between the 3d positions of the sender node and the receiver node computed in a local coordinate system of the receiver. The local coordinate system of the receiver is computed by applying a rotation that changes the azimuthal

angle until that receiver node lies at longitude 0, followed by a rotation that changes the polar angle until the receiver also lies at latitude 0. This results in a total of 327,660 mesh edges (See Table S3), each with 4 input features.

Grid2Mesh edges $\mathcal{E}^{\text{G2M}}$ are unidirectional edges that connect sender grid nodes to receiver mesh nodes. An edge $e^{\text{G2M}}_{v^{\text{G}}_{\text{s}} \to v^{\text{M}}_{\text{r}}}$ is added if the distance between the mesh node and the grid node is smaller or equal than 0.6 times[8] the length of the edges in mesh M^6 (see Figure 1) which ensures every grid node is connected to at least one mesh node. Features $\mathbf{e}^{\text{G2M,features}}_{v^{\text{G}}_{\text{s}} \to v^{\text{M}}_{\text{r}}}$ are built the same way as those for the mesh edges. This results on a total of 1,618,746 Grid2Mesh edges, each with 4 input features.

Mesh2Grid edges $\mathcal{E}^{\text{M2G}}$ are unidirectional edges that connect sender mesh nodes to receiver grid nodes. For each grid point, we find the triangular face in the mesh M^6 that contains it and add three Mesh2Grid edges of the form $e^{\text{M2G}}_{v^{\text{M}}_{\text{s}} \to v^{\text{G}}_{\text{r}}}$, to connect the grid node to the three mesh nodes adjacent to that face (see Figure 1). Features $\mathbf{e}^{\text{M2G,features}}_{v^{\text{M}}_{\text{s}} \to v^{\text{G}}_{\text{r}}}$ are built on the same way as those for the mesh edges. This results on a total of 3,114,720 Mesh2Grid edges (3 mesh nodes connected to each of the 721×1440 latitude-longitude grid points), each with four input features.

3.4 Encoder

The purpose of the encoder is to prepare data into latent representations for the processor, which will run exclusively on the multi-mesh.

Embedding the input features As part of the encoder, we first embed the features of each of the grid nodes, mesh nodes, mesh edges, grid to mesh edges, and mesh to grid edges into a

[8] Technically it is 0.6 times the "longest" edge in M^6, since there is some variance in the length of the edges caused by the split-and-reproject mechanism.

latent space of fixed size using five multi-layer perceptrons (MLP),

$$\begin{aligned}
\mathbf{v}_i^{\mathrm{G}} &= \mathrm{MLP}_{\mathcal{V}^{\mathrm{G}}}^{\text{embedder}}(\mathbf{v}_i^{\mathrm{G,features}}) \\
\mathbf{v}_i^{\mathrm{M}} &= \mathrm{MLP}_{\mathcal{V}^{\mathrm{M}}}^{\text{embedder}}(\mathbf{v}_i^{\mathrm{M,features}}) \\
\mathbf{e}_{v_{\mathrm{s}}^{\mathrm{M}} \to v_{\mathrm{r}}^{\mathrm{M}}}^{\mathrm{M}} &= \mathrm{MLP}_{\mathcal{E}^{\mathrm{M}}}^{\text{embedder}}(\mathbf{e}_{v_{\mathrm{s}}^{\mathrm{M}} \to v_{\mathrm{r}}^{\mathrm{M}}}^{\mathrm{M,features}}) \\
\mathbf{e}_{v_{\mathrm{s}}^{\mathrm{G}} \to v_{\mathrm{r}}^{\mathrm{M}}}^{\mathrm{G2M}} &= \mathrm{MLP}_{\mathcal{E}^{\mathrm{G2M}}}^{\text{embedder}}(\mathbf{e}_{v_{\mathrm{s}}^{\mathrm{G}} \to v_{\mathrm{r}}^{\mathrm{M}}}^{\mathrm{G2M,features}}) \\
\mathbf{e}_{v_{\mathrm{s}}^{\mathrm{M}} \to v_{\mathrm{r}}^{\mathrm{G}}}^{\mathrm{M2G}} &= \mathrm{MLP}_{\mathcal{E}^{\mathrm{M2G}}}^{\text{embedder}}(\mathbf{e}_{v_{\mathrm{s}}^{\mathrm{M}} \to v_{\mathrm{r}}^{\mathrm{G}}}^{\mathrm{M2G,features}})
\end{aligned} \tag{6}$$

Grid2Mesh GNN Next, in order to transfer information of the state of atmosphere from the grid nodes to the mesh nodes, we perform a single message passing step over the Grid2Mesh bipartite subgraph $\mathcal{G}_{\mathrm{G2M}}(\mathcal{V}^{\mathrm{G}}, \mathcal{V}^{\mathrm{M}}, \mathcal{E}^{\mathrm{G2M}})$ connecting grid nodes to mesh nodes. This update is performed using an interaction network (*17, 46*), augmented to be able to work with multiple node types (*47*). First, each of the Grid2Mesh edges are updated using information from the adjacent nodes,

$$\mathbf{e}_{v_{\mathrm{s}}^{\mathrm{G}} \to v_{\mathrm{r}}^{M}}^{\mathrm{G2M}\ \prime} = \mathrm{MLP}_{\mathcal{E}^{\mathrm{G2M}}}^{\mathrm{Grid2Mesh}}([\mathbf{e}_{v_{\mathrm{s}}^{\mathrm{G}} \to v_{\mathrm{r}}^{M}}^{\mathrm{G2M}}, \mathbf{v}_s^{\mathrm{G}}, \mathbf{v}_r^{M}]). \tag{7}$$

Then each of the mesh nodes is updated by aggregating information from all of the edges arriving at that mesh node:

$$\mathbf{v}_i^{\mathrm{M}\prime} = \mathrm{MLP}_{\mathcal{V}^{\mathrm{M}}}^{\mathrm{Grid2Mesh}}\Big(\Big[\mathbf{v}_i^{\mathrm{M}}, \sum_{e_{v_{\mathrm{s}}^{\mathrm{G}} \to v_{\mathrm{r}}^{\mathrm{M}}}^{\mathrm{G2M}}:\, v_{\mathrm{r}}^{\mathrm{M}} = v_i^{\mathrm{M}}} \mathbf{e}_{v_{\mathrm{s}}^{\mathrm{G}} \to v_{\mathrm{r}}^{\mathrm{M}}}^{\mathrm{G2M}\ \prime}\Big]\Big). \tag{8}$$

Each of the grid nodes are also updated, but with no aggregation, because grid nodes are not receivers of any edges in the Grid2Mesh subgraph,

$$\mathbf{v}_i^{\mathrm{G}\prime} = \mathrm{MLP}_{\mathcal{V}^{\mathrm{G}}}^{\mathrm{Grid2Mesh}}\big(\mathbf{v}_i^{\mathrm{G}}\big). \tag{9}$$

After updating all three elements, the model includes a residual connection, and for simplicity

of the notation, reassigns the variables,

$$\begin{aligned}
\mathbf{v}_i^{\mathrm{G}} &\leftarrow \mathbf{v}_i^{\mathrm{G}} + \mathbf{v}_i^{\mathrm{G}\prime}, \\
\mathbf{v}_i^{\mathrm{M}} &\leftarrow \mathbf{v}_i^{\mathrm{M}} + \mathbf{v}_i^{\mathrm{M}\prime}, \\
\mathbf{e}^{\mathrm{G2M}}_{v_{\mathrm{s}}^{\mathrm{G}} \to v_{\mathrm{r}}^{M}} &\leftarrow \mathbf{e}^{\mathrm{G2M}}_{v_{\mathrm{s}}^{\mathrm{G}} \to v_{\mathrm{r}}^{M}} + \mathbf{e}^{\mathrm{G2M}}_{v_{\mathrm{s}}^{\mathrm{G}} \to v_{\mathrm{r}}^{M}}{}'.
\end{aligned} \tag{10}$$

3.5 Processor

The processor is a deep GNN that operates on the Mesh subgraph $\mathcal{G}_{\mathrm{M}}(\mathcal{V}^{\mathrm{M}}, \mathcal{E}^{\mathrm{M}})$ which only contains the Mesh nodes and and the Mesh edges. Note the Mesh edges contain the full multi-mesh, with not only the edges of M^6, but all of the edges of M^5, M^4, M^3, M^2, M^1 and M^0, which will enable long distance communication.

Multi-mesh GNN A single layer of the Mesh GNN is a standard interaction network (*17, 46*) which first updates each of the mesh edges using information of the adjacent nodes:

$$\mathbf{e}^{\mathrm{M}}_{v_{\mathrm{s}}^{M} \to v_{\mathrm{r}}^{M}}{}' = \mathrm{MLP}^{\mathrm{Mesh}}_{\mathcal{E}^{\mathrm{M}}}([\mathbf{e}^{\mathrm{M}}_{v_{\mathrm{s}}^{M} \to v_{\mathrm{r}}^{M}}, \mathbf{v}_s^{M}, \mathbf{v}_r^{M}]). \tag{11}$$

Then it updates each of the mesh nodes, aggregating information from all of the edges arriving at that mesh node:

$$\mathbf{v}_i^{\mathrm{M}\prime} = \mathrm{MLP}^{\mathrm{Mesh}}_{\mathcal{V}^{\mathrm{M}}}\Big(\Big[\mathbf{v}_i^{\mathrm{M}}, \sum_{e^{\mathrm{M}}_{v_{\mathrm{s}}^{\mathrm{M}} \to v_{\mathrm{r}}^{\mathrm{M}}}:\, v_{\mathrm{r}}^{\mathrm{M}} = v_i^{\mathrm{M}}} \mathbf{e}^{\mathrm{M}}_{v_{\mathrm{s}}^{\mathrm{M}} \to v_{\mathrm{r}}^{\mathrm{M}}}{}'\Big]\Big) \tag{12}$$

And after updating both, the representations are updated with a residual connection and for simplicity of the notation, also reassigned to the input variables:

$$\begin{aligned}
\mathbf{v}_i^{\mathrm{M}} &\leftarrow \mathbf{v}_i^{\mathrm{M}} + \mathbf{v}_i^{\mathrm{M}\prime} \\
\mathbf{e}^{\mathrm{M}}_{v_{\mathrm{s}}^{M} \to v_{\mathrm{r}}^{M}} &\leftarrow \mathbf{e}^{\mathrm{M}}_{v_{\mathrm{s}}^{N} \to v_{\mathrm{r}}^{M}} + \mathbf{e}^{\mathrm{M}}_{v_{\mathrm{s}}^{M} \to v_{\mathrm{r}}^{M}}{}'
\end{aligned} \tag{13}$$

The previous paragraph describes a single layer of message passing, but following a similar approach to (*18, 19*), we used a stack of 16 layer, with unshared neural network weights for the MLPs in each layer. Note that while unshared across layers, within any given layer, the neural parameters of all of the MLPs in GraphCast are always shared across spatial locations.

3.6 Decoder

The role of the decoder is to bring back information to the grid, and extract an output.

Mesh2Grid GNN Analogous to the Grid2Mesh GNN, the Mesh2Grid GNN performs a single message passing over the Mesh2Grid bipartite subgraph $\mathcal{G}_{\text{M2G}}(\mathcal{V}^{\text{G}}, \mathcal{V}^{\text{M}}, \mathcal{E}^{\text{M2G}})$. The Grid2Mesh GNN is functionally equivalent to the Mesh2Grid GNN, but using the Mesh2Grid edges to send information in the opposite direction. The GNN first updates each of the Grid2Mesh edges using information of the adjacent nodes:

$$\mathbf{e}^{\text{M2G}}_{v^M_{\text{s}} \to v^{\text{G}}_{\text{r}}}{}' = \text{MLP}^{\text{Mesh2Grid}}_{\mathcal{E}^{\text{M2G}}}([\mathbf{e}^{\text{M2G}}_{v^M_{\text{s}} \to v^{\text{G}}_{\text{r}}}, \mathbf{v}^M_s, \mathbf{v}^{\text{G}}_r]) \tag{14}$$

Then it updates each of the grid nodes, aggregating information from all of the edges arriving at that grid node:

$$\mathbf{v}^{\text{G}\prime}_i = \text{MLP}^{\text{Mesh2Grid}}_{\mathcal{V}^{\text{G}}}\big([\mathbf{v}^{\text{G}}_i, \sum_{e^{\text{M2G}}_{v^{\text{M}}_{\text{s}} \to v^{\text{G}}_{\text{r}}}: v^{\text{G}}_{\text{r}} = v^{\text{G}}_i} \mathbf{e}^{\text{M2G}}_{v^{\text{M}}_{\text{s}} \to v^{\text{G}}_{\text{r}}}{}']\big). \tag{15}$$

In this case we do not update the mesh nodes, as they won't play any role from this point on.

Here again we add a residual connection, and for simplicity of the notation, reassign the variables, this time only for the grid nodes, which are the only ones required from this point on:

$$\mathbf{v}^{\text{G}}_i \leftarrow \mathbf{v}^{\text{G}}_i + \mathbf{v}^{\text{G}\prime}_i. \tag{16}$$

Output function Finally the prediction $\hat{\mathbf{y}}_i$ for each of the grid nodes is produced using another MLP,

$$\hat{\mathbf{y}}^{\text{G}}_i = \text{MLP}^{\text{Output}}_{\mathcal{V}^{\text{G}}}\big(\mathbf{v}^{\text{G}}_i\big) \tag{17}$$

which contains all 227 predicted variables for that grid node. Similar to (*18, 19*), the next weather state, $\hat{X}^{t+1}$, is computed by adding the per-node prediction, $\hat{Y}^t$, to the input state for

all grid nodes,

$$\hat{X}^{t+1} = \text{GraphCast}(X^t, X^{t-1}) = X^t + \hat{Y}^t. \tag{18}$$

3.7 Normalization and network parameterization

Input normalization Similar to (*18,19*), we normalized all inputs. For each physical variable, we computed the per-pressure level mean and standard deviation over 1979–2015, and used that to normalize them to zero mean and unit variance. For relative edge distances and lengths, we normalized the features to the length of the longest edge. For simplicity, we omit this output normalization from the notation.

Output normalization Because our model outputs a difference, $\hat{Y}^t$, which, during inference, is added to X^t to produce $\hat{X}^{t+1}$, we normalized the output of the model by computing per-pressure level standard deviation statistics for the time difference $Y^t = X^{t+1} - X^t$ of each variable[9]. When the GNN produces an output, we multiply this output by this standard deviation to obtain $\hat{Y}^t$ before computing $\hat{X}^{t+1}$, as in Equation (18). For simplicity, we omit this output normalization from the notation.

Neural network parameterizations The neural networks within GraphCast are all MLPs, with one hidden layer, and hidden and output layers sizes of 512 (except the final layer of the Decoder's MLP, whose output size is 227, matching the number of predicted variables for each grid node). Using a single hidden layer in MLPs is common practice in GNNs, as conventional wisdom, and our own experience, indicates increasing the number of layers of message-passing, and decreasing the depth of the MLPs, works best. Preliminary experiments of varied MLP depths supported this choice.

[9]We ignore the mean in the output normalization, as the mean of the time differences is zero.

We chose the "swish" (*48*) activation function for all MLPs. All MLPs are followed by a LayerNorm (*49*) layer (except for the Decoder's MLP).

4 Training details

This section provides details pertaining to the training of GraphCast, including the data split used to develop the model (Section 4.1), the full definition of the objective function with the weight associated with each variable and vertical level (Section 4.2), the autoregressive training approach (Section 4.3), optimization settings (Section 4.4), curriculum training used to reduce training cost (Section 4.5), technical details used to reduce the memory footprint of GraphCast (Section 4.6), training time (Section 4.7) and the software stacked we used (Section 4.8).

4.1 Training split

To mimic real deployment conditions, in which the forecast cannot depend on information from the future, we split the data used to develop GraphCast and data used to test its performance "causally", in that the "development set" only contained dates earlier than those in the "test set", as recommended in the WeatherBench benchmark (*8*). Similarly to (*14*), the development set comprises the period 1979–2017, and the test set contains the years 2018–2021. Neither the researchers, nor the model training software, were allowed to view data from the test set until we had finished the development phase. This prevented our choices of model architecture and training protocol from being able to exploit any information from the future.

Within our development set, we further split the data into a training set comprising the years 1979–2015, and a validation set that includes 2016–2017. We used the training set as training data for our models and the validation set for hyperparameter optimization and model selection, i.e., to decide on the best-performing model architecture. We then froze the model architecture and all the training choices and moved to the test phase. In preliminary work, we also explored training on earlier data from 1959–1978, but found it had little benefit on performance[10], so in

[10]We hypothesize this is due to the limited availability of observations, including satellite data, to construct ERA5 reanalysis prior to 1979 (*50*).

the final phases of our work we excluded 1959–1978 for simplicity.

4.2 Training objective

GraphCast was trained to minimize an objective function over 12-step forecasts (3 days) against ERA5 targets, using gradient descent. The training objective is defined as the mean square error (MSE) between the target output X and predicted output $\hat{X}$,

$$\mathcal{L}_{\text{MSE}} = \underbrace{\frac{1}{|D_{\text{batch}}|} \sum_{d_0 \in D_{\text{batch}}}}_{\text{forecast date-time}} \underbrace{\frac{1}{T_{\text{train}}} \sum_{\tau \in 1:T_{\text{train}}}}_{\text{lead time}} \underbrace{\frac{1}{|G_{0.25^\circ}|} \sum_{i \in G_{0.25^\circ}}}_{\text{spatial location}} \underbrace{\sum_{j \in J}}_{\text{variable-level}} s_j w_j a_i \underbrace{(\hat{x}_{i,j}^{d_0+\tau} - x_{i,j}^{d_0+\tau})^2}_{\text{squared error}} \quad (19)$$

where

- $\tau \in 1 : T_{\text{train}}$ are the lead times that correspond to the T_{train} autoregressive steps.
- $d_0 \in D_{\text{batch}}$ represent forecast initialization date-times in a batch of forecasts in the training set,
- $j \in J$ indexes the variable, and for atmospheric variables the pressure level. E.g. $J = \{\text{Z1000}, \text{Z850}, \ldots, \text{2T}, \text{MSL}\}$,
- $i \in G_{0.25^\circ}$ are the location (latitude and longitude) coordinates in the grid,
- $\hat{x}_{i,j}^{d_0+\tau}$ and $x_{i,j}^{d_0+\tau}$ are predicted and target values for some variable-level, location, and lead time,
- s_j is the per-variable-level inverse variance of time differences,
- w_j is the per-variable-level loss weight,
- a_i is the area of the latitude-longitude grid cell, which varies with latitude, and is normalized to unit mean over the grid.

In order to build a single scalar loss, we took the average across latitude-longitude, pressure levels, variables, lead times, and batch size. We averaged across latitude-longitude axes, with a weight proportional to the latitude-longitude cell size (normalized to mean 1). We applied uniform averages across time and batch.

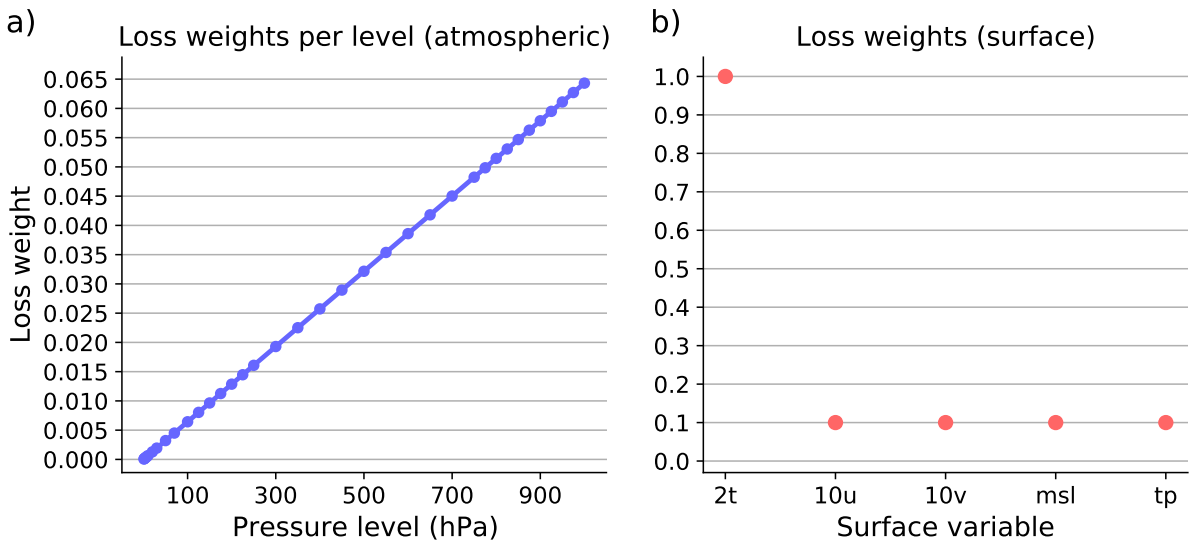

Fig. S2: **Training loss weights.** (a) Loss weights per pressure level, for atmospheric variables. (b) Loss weights for surface variables.

The quantities $s_j = \mathbb{V}_{i,t}\left[x_{i,j}^{t+1} - x_{i,j}^{t}\right]^{-1}$ are per-variable-level inverse variance estimates of the time differences, which aim to standardize the targets (over consecutive steps) to unit variance. These were estimated from the training data. We then applied per-variable-level loss weights, w_j. For atmospheric variables, we averaged across levels, with a weight proportional to the pressure of the level (normalized to unit mean), as shown in Figure S2a. We use pressure here as a proxy for the density (*13*). This is a design choice reflecting the higher importance we decided to assign to levels close to the surface. Note that the loss weight applied to pressure levels at or below 50 hPa, where HRES tends to perform better than GraphCast, is only 0.66% of the total loss weight across all variables and levels. We tuned the loss weights for the surface variables during model development, so as to produce roughly comparable validation performance across all variables: the weight on 2T was 1.0, and the weights on 10U, 10V, MSL, and TP were each 0.1, as shown in Figure S2b. The loss weights across all variables sum to 7.4, i.e., (6×1.0 for the atmospheric variables, plus $(1.0 + 0.1 + 0.1 + 0.1 + 0.1)$ for the surface variables listed above, respectively).

In follow up work, we found that by replacing the product of the vertical weight w_j and the inverse residual variance s_j by the per-level-per-variable weights defined as the inverse 6h-MSE of HRES, one could improve the performance of GraphCast at low pressure level for a number of variables, while maintaining the general performance advantage with respect to HRES. This suggests that there is no fundamental limitation to obtaining good performance at low pressure, and with the appropriate weighting, and modeling additional variables relevant to phenomenon occurring at low pressure, GraphCast could further improve performance above the tropopause.

4.3 Training on autoregressive objective

In order to improve our model's ability to make accurate forecasts over more than one step, we used an autoregressive training regime, where the model's predicted next step was fed back in as input for predicting the next step. The final GraphCast version was trained on 12 autoregressive steps, following a curriculum training schedule described below. The optimization procedure computed the loss on each step of the forecast, with respect to the corresponding ground truth step, error gradients with respect to the model parameters were backpropagated through the full unrolled sequence of model iterations (i.e., using backpropagation-through-time).

4.4 Optimization

The training objective function was minimized using gradient descent, with mini-batches. We sampled ground truth trajectories from our ERA5 training dataset, with replacement, for batches of size 32. We used the AdamW optimizer (*51,52*) with parameters $(\text{beta1} = 0.9, \text{beta2} = 0.95)$. We used weight decay of 0.1 on the weight matrices. We used gradient (norm) clipping with a maximum norm value of 32.

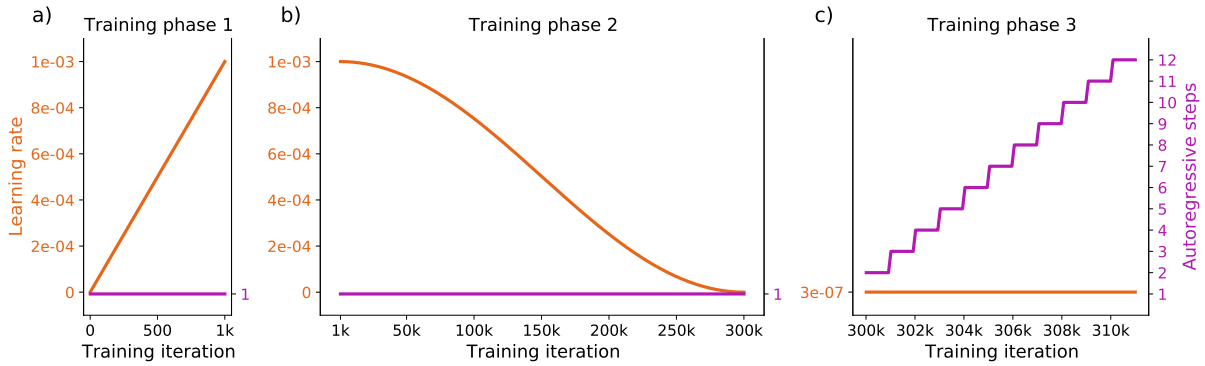

Fig. S3: **Training schedule.** (a) First phase of training. (b) Second phase of training. (c) Third phase of training. Left orange y-axis indicates the learning rate values. Right purple y-axis indicates the number of autoregressive steps used in the loss.

4.5 Curriculum training schedule

Training the model was conducted using a curriculum of three phases, which varied the learning rates and number of autoregressive steps. The first phase consisted of 1000 gradient descent updates, with one autoregressive step, and a learning rate schedule that increased linearly from 0 to $1\mathrm{e}{-3}$ (Figure S3a). The second phase consisted of 299,000 gradient descent updates, again with one autoregressive step, and a learning rate schedule that decreased back to 0 with half-cosine decay function (*53*) (Figure S3b). The third phase consisted of 11,000 gradient descent updates, where the number of autoregressive steps increased from 2 to 12, increasing by 1 every 1000 updates, and with a fixed learning rate of $3\mathrm{e}{-7}$ (Figure S3c).

The schedule for the number of autoregressive steps in the loss aims at reducing the computational cost of training by using a less expensive loss for most of the steps. By linearly increasing the number of steps in the autoregressive loss, the model focuses on obtaining good performance at short lead time first, a necessary step before learning how to reduce the error at long lead time.

4.6 Reducing memory footprint

To fit long trajectories (12 autoregressive steps) into the 32GB of a Cloud TPU v4 device[11], we use several strategies to reduce the memory footprint of our model. First, we use batch parallelism to distribute data across 32 TPU devices (i.e., one data point per device). Second, we use bfloat16 floating point precision to decrease the memory taken by activations (note, we use full-precision numerics (i.e. float32) to compute performance metrics at evaluation time). Finally, we use gradient check-pointing (*54*) to further reduce memory footprint at the cost of a lower training speed.

4.7 Training time

Following the training schedule that ramps up the number of autoregressive steps, as detailed above, training GraphCast took about four weeks on 32 TPU devices.

4.8 Software and hardware stack

We use JAX (*55*), Haiku (*56*), Jraph (*57*), Optax, Jaxline (*58*) and xarray (*59*) to build and train our models.

[11]For information about Cloud TPU v4 performance, memory and energy consumption, see https://cloud.google.com/tpu/docs/system-architecture-tpu-vm#tpu_v4.

5 Verification methods

This section provides details on our evaluation protocol. Section 5.1 details our approach to splitting data in a causal way, ensuring our evaluation tests for meaningful generalization, i.e., without leveraging information from the future. Section 5.2 explains in further details our choices to evaluate HRES skill and compare it to GraphCast, starting from the need for a ground truth specific to HRES to avoid penalizing it at short lead times (Section 5.2.1), the impact of ERA5 and HRES using different assimilation windows on the lookahead each state incorporates (Section 5.2.2), the resulting choice of initialization time for GraphCast and HRES to ensure that all methods benefit from the same lookahead in their inputs as well as in their targets (Section 5.2.3), and finally the evaluation period we used to report performance on 2018 (Section 5.2.4). Section 5.3 provides the definition of the metrics used to measure skill in our main results, as well as metrics used in complementary results in the Supplements. Finally, Section 5.4 details our statistical testing methodology.

5.1 Training, validation, and test splits

In the test phase, using protocol frozen at the end of the development phase (Section 4.1), we trained four versions of GraphCast, each of them on a different period. The models were trained on data from 1979–2017, 1979–2018, 1979–2019 and 1979–2020 for evaluation on the periods 2018–2021, 2019–2021, 2020–2021 and 2021, respectively. Again, these splits maintained a causal separation between the data used to train a version of the model and the data used to evaluate its performance (see Figure S4). Most of our results were evaluated on 2018 (i.e., with the model trained on 1979–2017), with several exceptions. For cyclone tracking experiments, we report results on 2018–2021 because cyclones are not that common, so including more years increases the sample size. We use the most recent version of GraphCast to make forecast on a

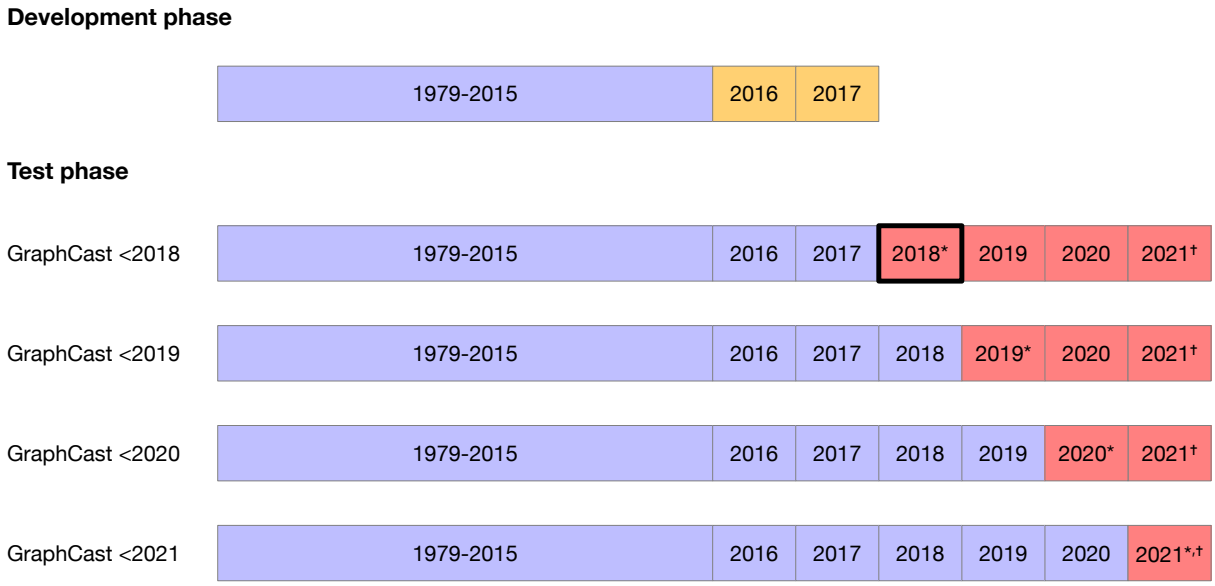

Fig. S4: **Data split summary.** In the development phase, GraphCast was trained on 1979–2015 (blue) and validated on 2016–2017 (yellow) until the training protocol was frozen. In the test phase, four versions of GraphCast were trained on larger and more recent train sets. Blue years represent training years for a given version of GraphCast, and red years represent the data that can be used at test time while satisfying split causality. Note that in the test phase, because the training protocol uses a fixed number of steps, we do not need validation data to compute a stopping criterion. The box represents the model and test year pair used in most of the results presented in the paper. The asterisks represent the model and test year pairs used in the cyclone tracking experiments. The dagger symbols denote the model and test year pairs used to characterize the effect of data recency.

given year: GraphCast <2018 for 2018 forecast, GraphCast <2019 for 2019 forecast, etc. For training data recency experiments, we evaluated how different models trained up to different years compared on 2021 test performance.

5.2 Comparing GraphCast to HRES

5.2.1 Choice of ground truth datasets

GraphCast was trained to predict ERA5 data, and to take ERA5 data as input; we also use ERA5 as ground truth for evaluating our model. HRES forecasts, however, are initialized based on HRES analysis. Generally, verifying a model against its own analysis gives the best skill estimates (*60*). So rather than evaluating HRES forecasts against ERA5 ground truth, which

would mean that even the zeroth step of HRES forecasts would have non-zero error, we constructed an "HRES forecast at step 0" (HRES-fc0) dataset, which contains the initial time step of HRES forecasts at future initializations (see Table S2). We use HRES-fc0 as ground truth for evaluating HRES forecasts.

5.2.2 Ensuring equal lookahead in assimilation windows

When comparing the skills of GraphCast and HRES, we made several choices to control for differences between the ERA5 and HRES-fc0 data assimilation windows. As described in Section 1, for each day of the test year 2018, HRES assimilates observations using four windows around 00z, 06z, 12z, and 18z (where 18z means 18:00 UTC in Zulu convention), each window including observations up to +3h after each corresponding time and date (see (*61*), Figure 11.2). However, ERA5 uses two windows around 00z and 12z assimilating observation up to +9h ahead, or equivalently two windows around 06z and 18z assimilating observation up to +3h ahead (see (*21*), Table 2). See Figure S5 for an illustration. We chose to evaluate GraphCast's forecasts from the 06z and 18z initializations, ensuring its inputs carry information from +3h of future observations, matching HRES's inputs. We did not evaluate GraphCast's 00z and 12z initializations, to avoid a mismatch between having a +9h lookahead in ERA5 inputs versus +3h lookahead for HRES inputs. Figure S6 and Figure S7 show the performance of GraphCast initialized from 06z/18z, and 00z/12z. When initialized from a state with a larger lookahead, GraphCast gets a visible improvement that persists at longer lead times, supporting our choice to initialized evaluation from 06z/18z.

We applied the same logic when choosing the target on which to evaluate: we only evaluate targets which incorporate a +3h lookahead for both HRES and ERA5. Given our choice of initialization at 06z and 18z, this corresponds to evaluating every 12h, on future 06z and 18z analysis times. As a practical example, if we were to evaluate GraphCast and HRES initialized

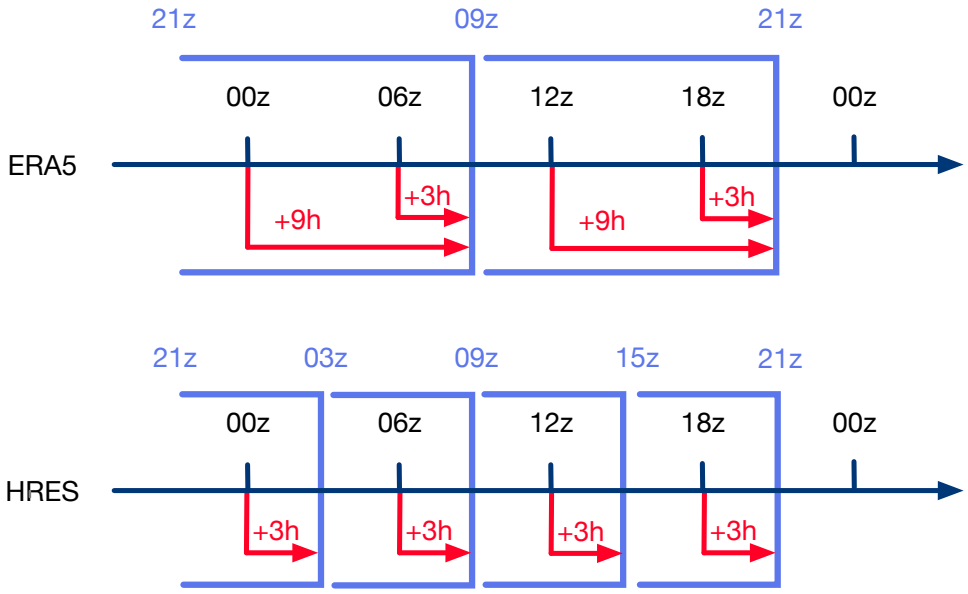

Fig. S5: **Schematic of the assimilation windows for ERA5 and HRES.** Data assimilation windows are marked as blue open rectangles. The red arrows represent the duration of the effective lookahead that is incorporated in the corresponding state.

at 06z, at lead time 6h (i.e., 12z), the target for GraphCast would integrate a +9h lookahead, while the target for HRES would only incorporate +3h lookahead. At equal lead time, this could result in a harder task for GraphCast.

5.2.3 Alignment of initialization and validity times-of-day

As stated above, a fair comparison with HRES requires us to evaluate GraphCast using 06z and 18z initializations, and with lead times which are multiples of 12h, meaning validity times are also 06z and 18z.

For lead times up to 3.75 days there are archived HRES forecasts available using 06z and 18z initialization and validity times, and we use these to perform a like-for-like comparison with GraphCast at these lead times. Note, because we evaluate only on 12 hour lead time increments, this means the final lead time is 3.5 days.

For lead times of 4 days and beyond, archived HRES forecasts are only available at 00z and

12z initializations, which given our 12-hour-multiple lead times means 00z and 12z validity times. At these lead times we have no choice but to compare GraphCast at 06z and 18z, with HRES at 00z and 12z.

In Figure S8, we can see that up to 3.5 day lead times, HRES RMSEs tend to be smaller on average over 00z and 12z initialization/validity times than they are at the 06z and 18z times which GraphCast is evaluated on. We can also see that the difference decreases as lead time increases, and that the 06z/18z RMSEs generally appear to be tending towards an asymptote above the 00z/12z RMSE, but within 2% of it. We expect these differences to remain small, and so we do not believe that they compromise our conclusions in cases where GraphCast has greater skill than HRES.

Whenever we plot RMSE and other evaluation metrics as a function of lead time, we indicate with a dotted line the 3.5 day changeover point where we switch from evaluating HRES on 06z/18z to evaluating on 00z/12z. At this changeover point, we plot both the 06z/18z and 00z/12z metrics, showing the discontinuity clearly.

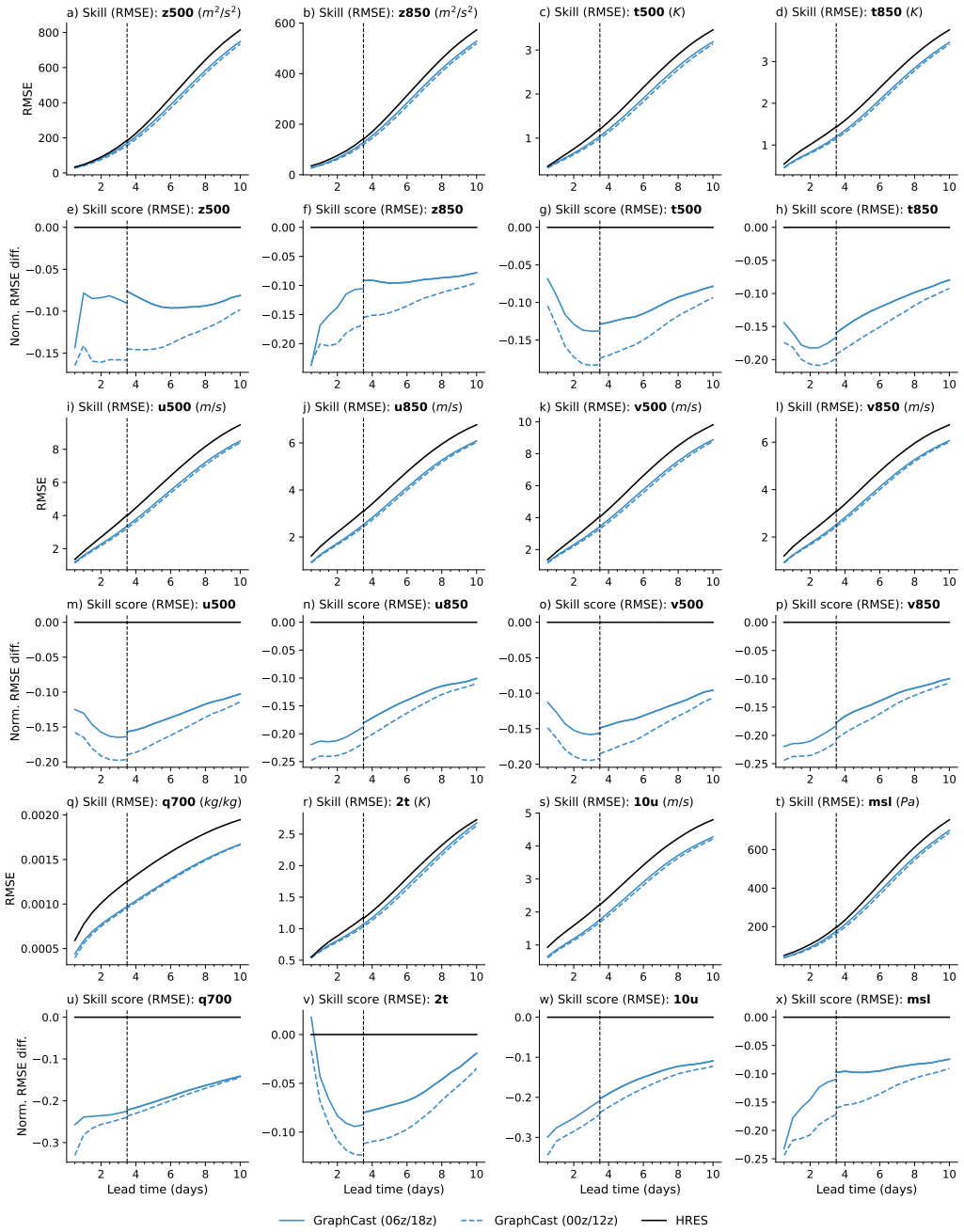

Fig. S6: **Effect of initialization time for GraphCast only.** This figure is analogous to Figure 2a,b, as well as to Figure S10, but showing results separately for GraphCast initialized at 00z/12z vs 06z/18z, while using the same baseline we used in the main results (HRES 06z/18z as baseline up to 3.5 days and HRES 00z/12z beyond 3.5 days). When initialized from an ERA5 state benefiting from longer lookahead (00z/12z), GraphCast performs better than when initialized from an ERA5 state benefiting from a shorter lookahead (06z/18z). The improvement is measurable from short to long prediction lead time. This supports our choice to evaluate all models from 06z/18z, in order to avoid giving an artificial advantage to GraphCast. The x-axis represents lead time, at 12-hour steps over 10 days. The y-axis represents the RMSE skill or skill score.

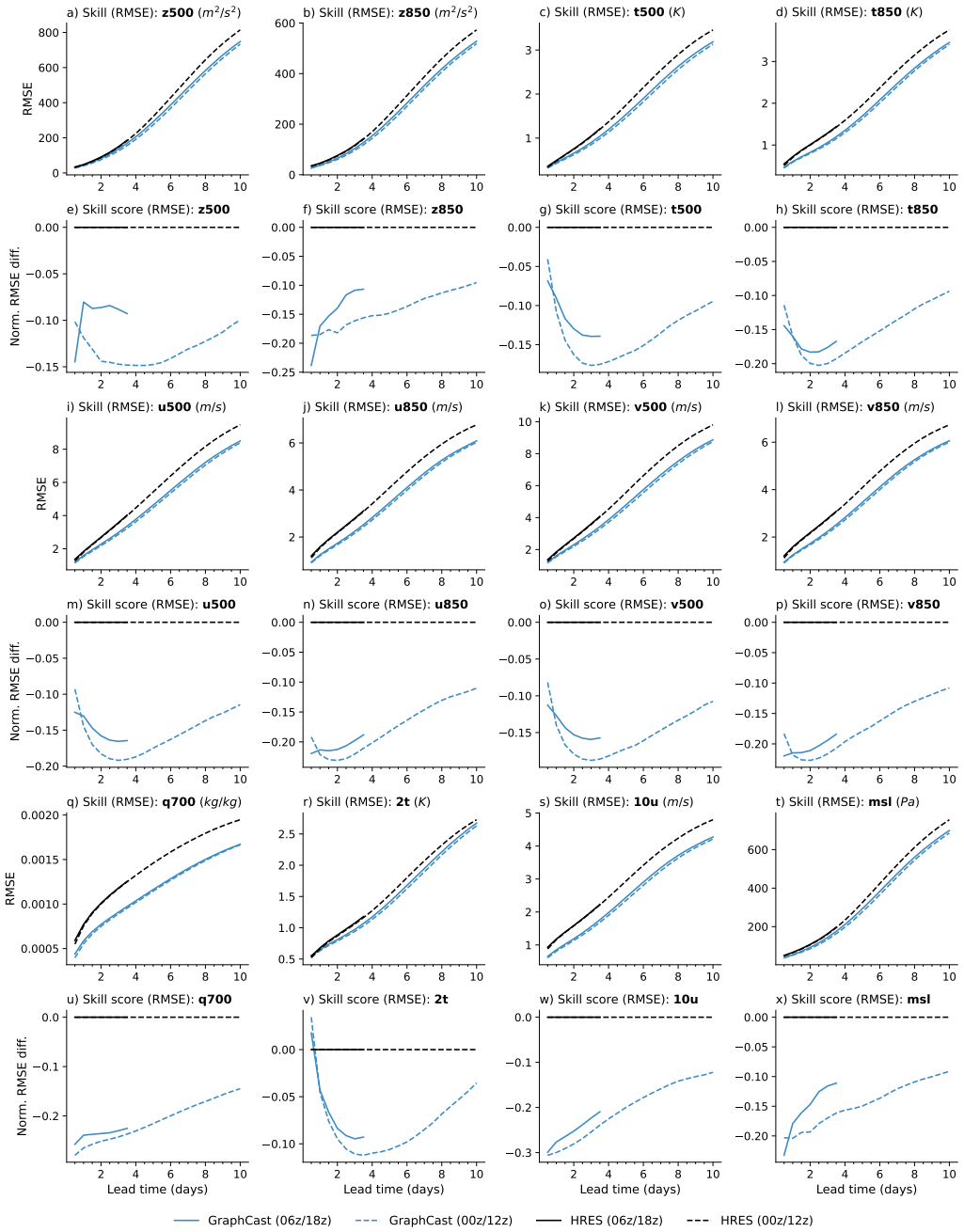

Fig. S7: **Effect of initialization time.** This figure is analogous to Figure 2a,b, as well as to Figure S10, but showing results separately for 00z/12z and 06z/18z initializations, such that GraphCast 00z/12z is compared to HRES 00z/12z, and GraphCast 06z/18z is compared to HRES 06z/18z. When initializing from 00z/12z states with benefit from an extra ERA5 lookahead, GraphCast performs better than when initialized from 06z/18z states which has the same lookahead for ERA5 and HRES-fc0. The improvement is measurable from short to long prediction lead time. This supports our choice to evaluate all models from 06z/18z when possible, in order to avoid giving an artificial advantage to GraphCast. The x-axis represents lead time, at 12-hour steps over 10 days. The y-axis represents the RMSE skill or skill score.

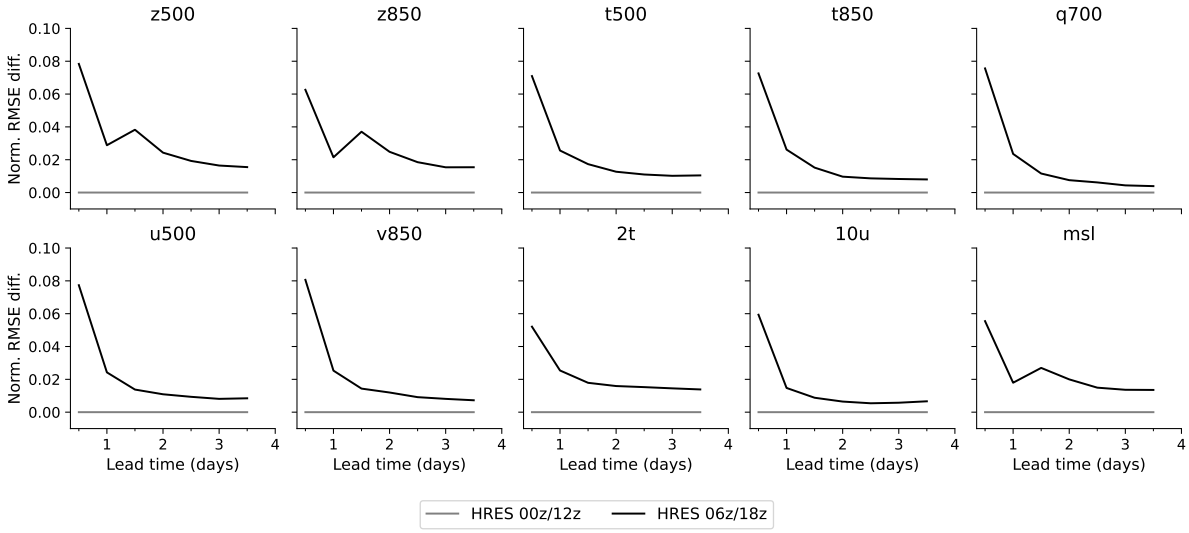

Fig. S8: **RMSE skill scores for HRES at 06z/18z vs HRES at 00z/12z.** Plots show the RMSE skill score of HRES initialized at 06z/18z (black line), relative to HRES initialized at 00z/12z (grey line), as a function of lead time. We plot lead times up to 3.5 days which is the longest for which HRES predictions initialized at 06z/18z are available. The y axis scales are shared.

5.2.4 Evaluation period

Most of our main results are reported for the year 2018 (from our test set), for which the first forecast initialization time was `2018-01-01_06:00:00` UTC and the last `2018-12-31_18:00:00`, or when evaluating HRES at longer lead times, `2018-01-01_00:00:00` and `2018-12-31_12:00:00`. Additional results on cyclone tracking and the effect of data recency use years 2018–2021 and 2021 respectively.

5.3 Evaluation metrics

We quantify the skillfulness of GraphCast, other ML models, and HRES using the root mean square error (RMSE) and the anomaly correlation coefficient (ACC), which are both computed against the models' respective ground truth data. The RMSE measures the magnitude of the differences between forecasts and ground truth for a given variable indexed by j and a given lead time τ (see Equation (20)). The ACC, $\mathcal{L}_{\text{ACC}}^{j,\tau}$, is defined in Equation (28) and measures the correlation between forecasts' differences from climatology and ground truth's differences from climatology, where climatology is the average weather for a location and date. For skill scores we use the normalized RMSE difference between model A and baseline B as $(\text{RMSE}_A - \text{RMSE}_B)/\text{RMSE}_B$, and the normalized ACC difference as $(\text{ACC}_A - \text{ACC}_B)/(1 - \text{ACC}_B)$.

All metrics were computed using float32 precision and reported using the native dynamic range of the variables, without normalization.

Root mean square error (RMSE). We quantified forecast skill for a given variable, x_j, and lead time, $\tau = t\Delta d$, using a latitude-weighted root mean square error (RMSE) given by

$$\text{RMSE}(j,\tau) = \frac{1}{|D_{\text{eval}}|} \sum_{d_0 \in D_{\text{eval}}} \sqrt{\frac{1}{|G_{0.25^\circ}|} \sum_{i \in G_{0.25^\circ}} a_i \left(\hat{x}_{j,i}^{d_0+\tau} - x_{j,i}^{d_0+\tau}\right)^2} \tag{20}$$

where

- $d_0 \in D_{\text{eval}}$ represent forecast initialization date-times in the evaluation dataset,
- $j \in J$ index variables and levels, e.g., $J = \{\text{Z1000}, \text{Z850}, \ldots, \text{2T}, \text{MSL}\}$,
- $i \in G_{0.25^\circ}$ are the location (latitude and longitude) coordinates in the grid,
- $\hat{x}_{j,i}^{d_0+\tau}$ and $x_{j,i}^{d_0+\tau}$ are predicted and target values for some variable-level, location, and lead time,
- a_i is the area of the latitude-longitude grid cell (normalized to unit mean over the grid) which varies with latitude.

By taking the square root inside the mean over forecast initializations we follow the convention of WeatherBench (*8*). However we note that this differs from how RMSE is defined in many other contexts, where the square root is only applied to the final mean, that is,

$$\text{RMSE}_{\text{trad}}(j,\tau) = \sqrt{\frac{1}{|D_{\text{eval}}|}\sum_{d_0 \in D_{\text{eval}}} \frac{1}{|G_{0.25^\circ}|}\sum_{i \in G_{0.25^\circ}} a_i \left(\hat{x}_{j,i}^{d_0+\tau} - x_{j,i}^{d_0+\tau}\right)^2}. \tag{21}$$

Root mean square error (RMSE), spherical harmonic domain. In all comparisons involving predictions that are filtered, truncated or decomposed in the spherical harmonic domain, for convenience we compute RMSEs directly in the spherical harmonic domain, with all means taken inside the square root,

$$\text{RMSE}_{\text{sh}}(j,\tau) = \sqrt{\frac{1}{|D_{\text{eval}}|}\sum_{d_0 \in D_{\text{eval}}} \frac{1}{4\pi}\sum_{l=0}^{l_{max}}\sum_{m=-l}^{l}\left(\hat{f}_{j,l,m}^{d_0+\tau} - f_{j,l,m}^{d_0+\tau}\right)^2} \tag{22}$$

Here $\hat{f}_{j,l,m}^{d_0+\tau}$ and $f_{j,l,m}^{d_0+\tau}$ are predicted and target coefficients of spherical harmonics with total wavenumber l and longitudinal wavenumber m. We compute these coefficients from grid-based data using a discrete spherical harmonic transform (*62*) with triangular truncation at wavenumber 719, which was chosen to resolve the 0.25° (28km) resolution of our grid at the equator. This means that l ranges from 0 to $l_{max} = 719$ and m from $-l$ to l. We chose to plot this full range, rather than truncating at ERA5's highest wavenumber of 639, and note that regridding from 639 to 0.25° leads to some content above 639 as well.

This RMSE closely approximates the grid-based definition of RMSE given in Equation (21), however it is not exactly comparable, in part because the triangular truncation at wavenumber 719 does not resolve the additional resolution of the equiangular grid near the poles.

Root mean square error (RMSE), per location. This is computed following the RMSE definition of Equation (21), but for a single location:

$$\text{RMSE}_{\text{by-lat-lon}}(i,j,\tau) = \sqrt{\frac{1}{|D_{\text{eval}}|}\sum_{d_0\in D_{\text{eval}}}\left(\hat{x}_{j,i}^{d_0+\tau} - x_{j,i}^{d_0+\tau}\right)^2}. \tag{23}$$

We also break down RMSE by latitude only:

$$\text{RMSE}_{\text{by-lat}}(l,j,\tau) = \sqrt{\frac{1}{|D_{\text{eval}}|}\sum_{d_0\in D_{\text{eval}}}\frac{1}{|\text{lon}(G_{0.25^\circ})|}\sum_{i\in G_{0.25^\circ}:\text{lat}(i)=l}\left(\hat{x}_{j,i}^{d_0+\tau} - x_{j,i}^{d_0+\tau}\right)^2} \tag{24}$$

where $|\text{lon}(G_{0.25^\circ})| = 1440$ is the number of distinct longitudes in our regular 0.25° grid.

Root mean square error (RMSE), by surface elevation. This is computed following the RMSE definition of Equation (21) but restricted to a particular range of surface elevations, given by bounds $z_l \le z_{\text{surface}} < z_u$ on the surface geopotential:

$$\text{RMSE}_{\text{by-elevation}}(z_l,z_u,j,\tau) = \sqrt{\frac{\sum_{d_0\in D_{\text{eval}}}\sum_{i\in G_{0.25^\circ}}\mathbb{I}[z_l \le z_{surface}(i) < z_u]a_i(\hat{x}_{j,i}^{d_0+\tau} - x_{j,i}^{d_0+\tau})^2}{|D_{\text{eval}}|\sum_{i\in G_{0.25^\circ}}\mathbb{I}[z_l \le z_{surface}(i) < z_u]a_i}}, \tag{25}$$

where $\mathbb{I}$ denotes the indicator function.

Mean bias error (MBE), per location. This quantity is defined as

$$\text{MBE}_{\text{by-lat-lon}}(i,j,\tau) = \frac{1}{|D_{\text{eval}}|}\sum_{d_0\in D_{\text{eval}}}\left(\hat{x}_{j,i}^{d_0+\tau} - x_{j,i}^{d_0+\tau}\right). \tag{26}$$

Root-mean-square per-location mean bias error (RMS-MBE). This quantifies the average magnitude of the per-location biases from Equation (26) and is given by

$$\text{RMS-MBE}(j,\tau) = \sqrt{\frac{1}{|G_{0.25^\circ}|}\sum_{i\in G_{0.25^\circ}} a_i\ \text{MBE}_{\text{by-lat-lon}}(i,j,\tau)^2}. \tag{27}$$

Anomaly correlation coefficient (ACC). We also computed the anomaly correlation coefficient for a given variable, x_j, and lead time, $\tau = t\Delta d$, according to

$$\mathcal{L}_{\text{ACC}}^{j,\tau} = \frac{1}{|D_{\text{eval}}|} \sum_{d_0 \in D_{\text{eval}}} \frac{\sum_{i \in G_{0.25^\circ}} a_i \left(\hat{x}_{j,i}^{d_0+\tau} - C_{j,i}^{d_0+\tau}\right)\left(x_{j,i}^{d_0+\tau} - C_{j,i}^{d_0+\tau}\right)}{\sqrt{\left[\sum_{i \in G_{0.25^\circ}} a_i \left(\hat{x}_{j,i}^{d_0+\tau} - C_{j,i}^{d_0+\tau}\right)^2\right]\left[\sum_{i \in G_{0.25^\circ}} a_i \left(x_{j,i}^{d_0+\tau} - C_{j,i}^{d_0+\tau}\right)^2\right]}} \tag{28}$$

where $C_{j,i}^{d_0+\tau}$ is the climatological mean for a given variable, level, latitude and longitude, and for the day-of-year containing the validity time $d_0 + \tau$. Climatological means were computed using ERA5 data between 1993 and 2016. All other variables are defined as above.

5.4 Statistical methodology

5.4.1 Significance tests for difference in means

For each lead time τ and variable-level j, we test for a difference in means between per-initialization-time RMSEs (defined in Equation (29)) for GraphCast and HRES. We use a paired two-sided t-test with correction for auto-correlation, following the methodology of (*63*). This test assumes that time series of differences in forecast scores are adequately modelled as stationary Gaussian AR(2) processes. This assumption does not hold exactly for us, but is motivated as adequate for verification of medium range weather forecasts by the ECMWF in (*63*).

The nominal sample size for our tests is $n = 730$ at lead times under 4 days, consisting of two forecast initializations per day over the 365 days of 2018. (For lead times over 4 days we have $n = 729$, see Section 5.4.2). However these data (differences in forecast RMSEs) are auto-correlated in time. Following (*63*) we estimate an inflation factor k for the standard error which corrects for this. Values of k range between 1.21 and 6.75, with the highest values generally seen at short lead times and at the lowest pressure levels. These correspond to reduced effective sample sizes $n_{\text{eff}} = n/k^2$ in the range of 16 to 501.

See Table S4 for detailed results of our significance tests, including p-values, values of the t

test statistic and of n_{eff}.

5.4.2 Forecast alignment

For lead times τ less than 4 days, we have forecasts available at 06z and 18z initialization and validity times each day for both GraphCast and HRES, and we can test for differences in RMSEs between these paired forecasts. Defining the per-initialization-time RMSE as:

$$\text{RMSE}(j, \tau, d_0) = \sqrt{\frac{1}{|G_{0.25^\circ}|} \sum_{i \in G_{0.25^\circ}} a_i \left(\hat{x}_{j,i}^{d_0+\tau} - x_{j,i}^{d_0+\tau}\right)^2} \tag{29}$$

We compute differences

$$\text{diff-RMSE}(j, \tau, d_0) = \text{RMSE}_{GC}(j, \tau, d_0) - \text{RMSE}_{HRES}(j, \tau, d_0), \tag{30}$$

which we use to test the null hypothesis that $\mathbb{E}[\text{diff-RMSE}(j, \tau, d_0)] = 0$ against the two-sided alternative. Note that by our stationarity assumption this expectation does not depend on d_0.

As discussed in Section 5.2.3, at lead times of 4 days or more we only have HRES forecasts available at 00z and 12z initialization and validity times, while for the fairest comparison (Section 5.2.2) GraphCast forecasts must be evaluated using 06z and 18z initialization and validity times. In order to perform a paired test, we compare the RMSE of a GraphCast forecast with an interpolated RMSE of the two HRES forecasts either side of it: one initialized and valid 6 hours earlier, and the other initialized and valid 6 hours later, all with the same lead time. Specifically we compute differences:

$$\begin{aligned} \text{diff-RMSE}_{\text{interp}}(j, \tau, d_0) = \text{RMSE}_{GC}(j, \tau, d_0) \\ - \frac{1}{2}\Big(\text{RMSE}_{HRES}(j, \tau, d_0 - 6h) + \text{RMSE}_{HRES}(j, \tau, d_0 + 6h)\Big). \end{aligned} \tag{31}$$

We can use these to test the null hypothesis $\mathbb{E}[\text{diff-RMSE}_{\text{interp}}(j, \tau, d_0)] = 0$, which again doesn't depend on d_0 by the stationarity assumption on the differences. If we further assume that the HRES RMSE time series itself is stationary (or at least close enough to stationary

over a 6 hour window) then $\mathbb{E}[\text{diff-RMSE}_{\text{interp}}(j, \tau, d_0)] = \mathbb{E}[\text{diff-RMSE}(j, \tau, d_0)]$ and the interpolated differences can also be used to test deviations from the original null hypothesis that $\mathbb{E}[\text{diff-RMSE}(j, \tau, d_0)] = 0$.

This stronger stationarity assumption for HRES RMSEs is violated by diurnal periodicity, and in Section 5.2.3 we do see some systematic differences in HRES RMSEs between 00z/12z and 06z/18z validity times. However as discussed there, these systematic differences reduce substantially as lead time grows and they tend to favour HRES, and so we believe that a test of $\mathbb{E}[\text{diff-RMSE}(j, \tau, d_0)] = 0$ based on $\text{diff-RMSE}_{\text{interp}}$ will be conservative in cases where GraphCast appears to have greater skill than HRES.

5.4.3 Confidence intervals for RMSEs

The error bars in our RMSE skill plots correspond to separate confidence intervals for $\mathbb{E}[\text{RMSE}_{GC}]$ and $\mathbb{E}[\text{RMSE}_{HRES}]$ (eliding for now the arguments j, τ, d_0). These are derived from the two-sided t-test with correction for auto-correlation that is described above, applied separately to GraphCast and HRES RMSE time-series.

These confidence intervals make a stationarity assumption for the separate GraphCast and HRES RMSE time series, which as stated above is a stronger assumption that stationarity of the differences and is violated somewhat. Thus these single-sample confidence intervals should be treated as approximate; we do not rely on them in our significance statements.

5.4.4 Confidence intervals for RMSE skill scores

From the t-test described in Section 5.4.1 we can also derive in the standard way confidence intervals for the true difference in RMSEs, however in our skill score plots we would like to show confidence intervals for the true RMSE skill score, in which the true difference is

normalized by the true RMSE of HRES:

$$\text{RMSE-SS}_{\text{true}} = \frac{\mathbb{E}[\text{RMSE}_{GC} - \text{RMSE}_{HRES}]}{\mathbb{E}[\text{RMSE}_{HRES}]} \tag{32}$$

A confidence interval for this quantity should take into account the uncertainty of our estimate of the true HRES RMSE. Let $[l_{\text{diff}}, u_{\text{diff}}]$ be our $1 - \alpha/2$ confidence interval for the numerator (difference in RMSEs), and $[l_{\text{HRES}}, u_{\text{HRES}}]$ our $1 - \alpha/2$ confidence interval for the denominator (HRES RMSE). Given that $0 < l_{HRES}$ in every case for us, using interval arithmetic and the union bound we obtain a conservative $1 - \alpha$ confidence interval

$$[\min\{l_{\text{diff}}/u_{\text{HRES}}, l_{\text{diff}}/l_{\text{HRES}}\}, \max\{u_{\text{diff}}/u_{\text{HRES}}, u_{\text{diff}}/l_{\text{HRES}}, \}] \tag{33}$$

for $\text{RMSE-SS}_{\text{true}}$. We plot these confidence intervals alongside our estimates of the RMSE skill score, however note that we don't rely on them for significance testing.

Supplementary Text

6 Comparison with previous machine learning baselines

To determine how GraphCast's performance compares to other ML methods, we focus on Pangu-Weather (*16*), a strong MLWP baseline that operates at 0.25° resolution. To make the most direct comparison, we depart from our evaluation protocol, and use the one described in (*16*). Because published Pangu-Weather results are obtained from the 00z/12z initializations, we use those same initializations for GraphCast, instead of 06z/18z, as in the rest of this paper. This allows both models to be initialized on the same inputs, which incorporate the same amount of lookahead (+9 hours, see Sections 5.2.2 and 5.2.3). As HRES initialization incorporates at most +3 hours lookahead, even if initialized from 00z/12z, we do not show the evaluation of HRES (against ERA5 or against HRES-fc0) in this comparison as it would disadvantage it. The second difference with our protocol is to report performance every 6 hours, rather than every 12 hours. Since both models are evaluated against ERA5, their targets are identical, in particular, for a given lead time, the target incorporates +3 hours or +9 hours of lookahead for both GraphCast and Pangu-Weather, allowing for a fair comparison. Pangu-Weather (*16*) reports its 7-day forecast accuracy (RMSE and ACC) on: Z500, T500, T850, Q500, U500, V500, 2T, 10U, 10V, and MSL.

As shown in Figure S9, GraphCast (blue lines) outperforms Pangu-Weather (*16*) (red lines) on 99.2% of targets. For the surface variables (2T, 10U, 10V, MSL), GraphCast's error in the first several days is around 10-20% lower, and over the longer lead times plateaus to around 7-10% lower error. The only two (of the 252 total) metrics on which Pangu-Weather outperformed GraphCast was Z500, at lead times 6 and 12 hours, where GraphCast had 1.7% higher average RMSE (Figure S9a,e).

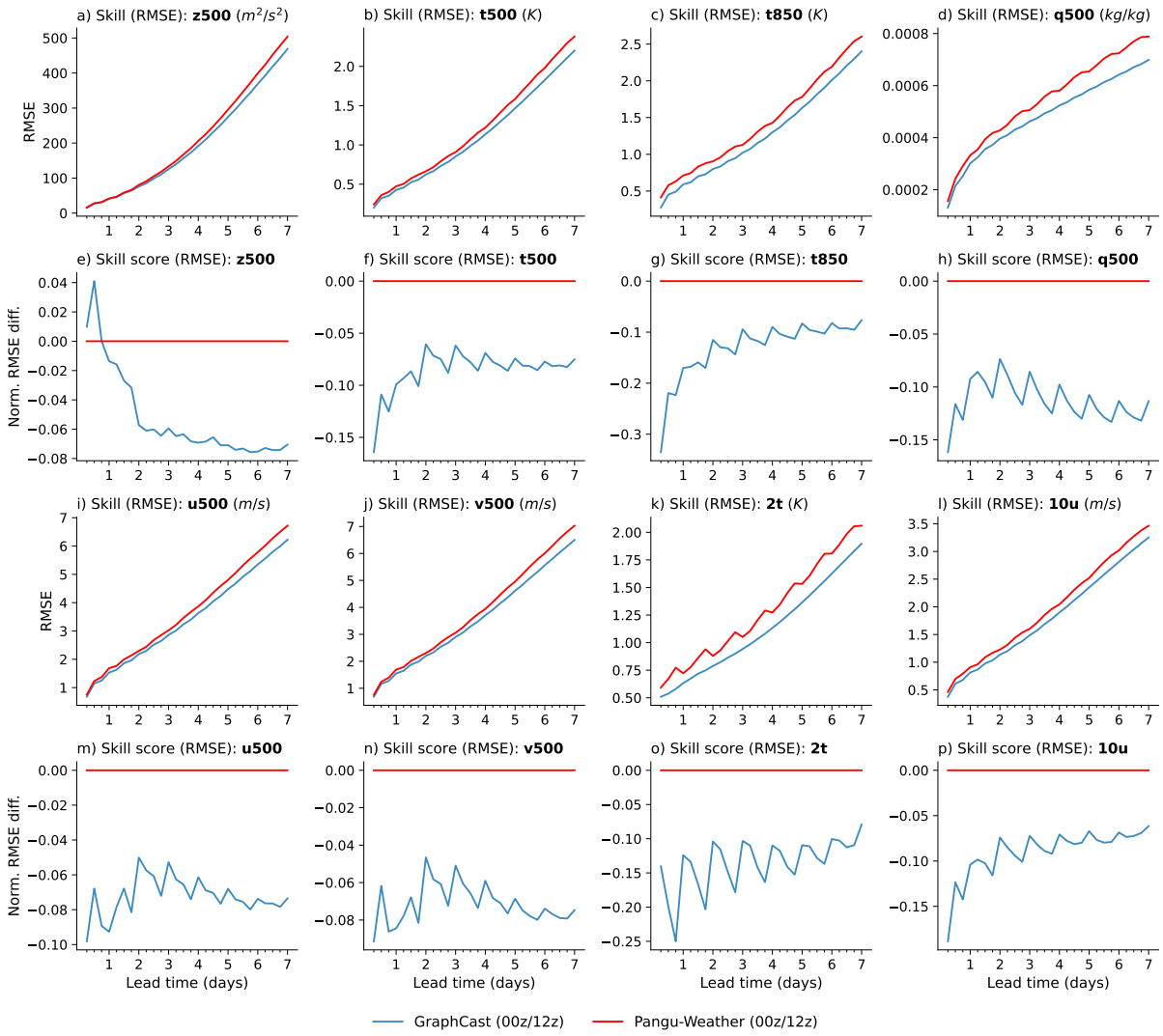

Fig. S9: **Comparison between GraphCast and Pangu-Weather, on RMSE skill.** Rows 1 and 3 show absolute RMSE for GraphCast (blue lines), and Pangu-Weather (*16*) (red lines); rows 2 and 4 show normalized RMSE differences between the models with respect to Pangu-Weather. Each subplot represents a single variable (and pressure level, for atmospheric variables), as indicated in the subplot titles. The x-axis represents lead time, at 6-hour steps over 10 days. The y-axis represents (absolute or normalized) RMSE. The variables and levels were chosen to be those reported by (*16*). We did not include 10V, because 10U is already present, and the two are highly correlated.

7 Additional forecast verification results

This section provides additional analysis of GraphCast's performance, giving a fuller picture of its strengths and limitations. Section 7.1 complements the main results of the paper on additional variables and levels beyond z500. Section 7.2 further analyses GraphCast performance broken down by regions, latitude and pressure levels (in particular distinguishing the performance below and above the tropopause), illustrates the biases and the RMSE by latitude longitude and elevation. Section 7.3 demonstrates that both the multi-mesh and the autoregressive loss play an important role in the performance of GraphCast. Section 7.4 details the approach of optimized blurring applied to HRES and GraphCast, to ensure that GraphCast improved performance is not only due to its ability to blur its predictions. It also shows the connection between the number of autoregressive steps in the loss and blurring, demonstrating that autoregressive training does more than just teaching the model to blur its predictions. Finally, Section 7.5 shows various spectral analyses, demonstrating that in most cases GraphCast has improved performance over HRES across all horizontal length scales and resolutions. We also discuss the impact of differences in spectra between ERA5 and HRES. Together, those results show an extensive evaluation of GraphCast and a rigorous comparison to HRES.

7.1 Detailed results for additional variables

7.1.1 RMSE and ACC

Figure S10 complements Figure 2a–b and shows the RMSE and normalized RMSE difference with respect to HRES for GraphCast and HRES on a combination of 12 highlight variables. Figure S11 shows the ACC and normalized ACC difference with respect to HRES for GraphCast and HRES on the same a combination of 12 variables and complements Figure 2c. The ACC skill score is the normalized ACC difference between model A and baseline B as $(\mathrm{ACC}_A -$

$\mathrm{ACC}_B)/(1 - \mathrm{RMSE}_B)$.

7.1.2 Detailed significance test results for RMSE comparisons

Table S4 provides further information about the statistical significance claims made in the main section about differences in RMSE between GraphCast and HRES. Details of the methodology are in Section 5.4. Here we give p-values, test statistics and effective sample sizes for all variables. For reasons of space we limit ourselves to three key lead times (12 hours, 2 days and 10 days) and a subset of 7 pressure levels chosen to include all cases where $p > 0.05$ at these lead times.

7.1.3 Effect of data recency on GraphCast

An important feature of MLWP methods is that a new model can be trained periodically with the most recent data. This, in principle, allows them to model recent weather patterns that change over time, as well as the effects of climate change. To explore how the recency of the training data influences GraphCast's test performance, we trained four variants of GraphCast, from scratch, with training data that always began in 1979, but ended in 2017, 2018, 2019, and 2020, respectively (we label the variant ending in 2017 as "GraphCast:<2018", etc). We evaluated the variants, and HRES, on 2021 test data.

Figure S12 shows the skill and skill scores (with respect to HRES) of the four variants of GraphCast, for several variables and complements Figure 4a. There is a general trend where variants trained to years closer to the test year have generally improved skill score against HRES. The reason for this improvement is not fully understood, though we speculate it is analogous to bias correction, but over longer time scales, where statistical biases or changes in the weather are being exploited to improve accuracy. It is also important to note that HRES is not a single NWP across years: it tends to be upgraded once or twice a year, with generally increasing skill on z500 and other fields (*64–68*). This may also contribute to why GraphCast:<2018 and

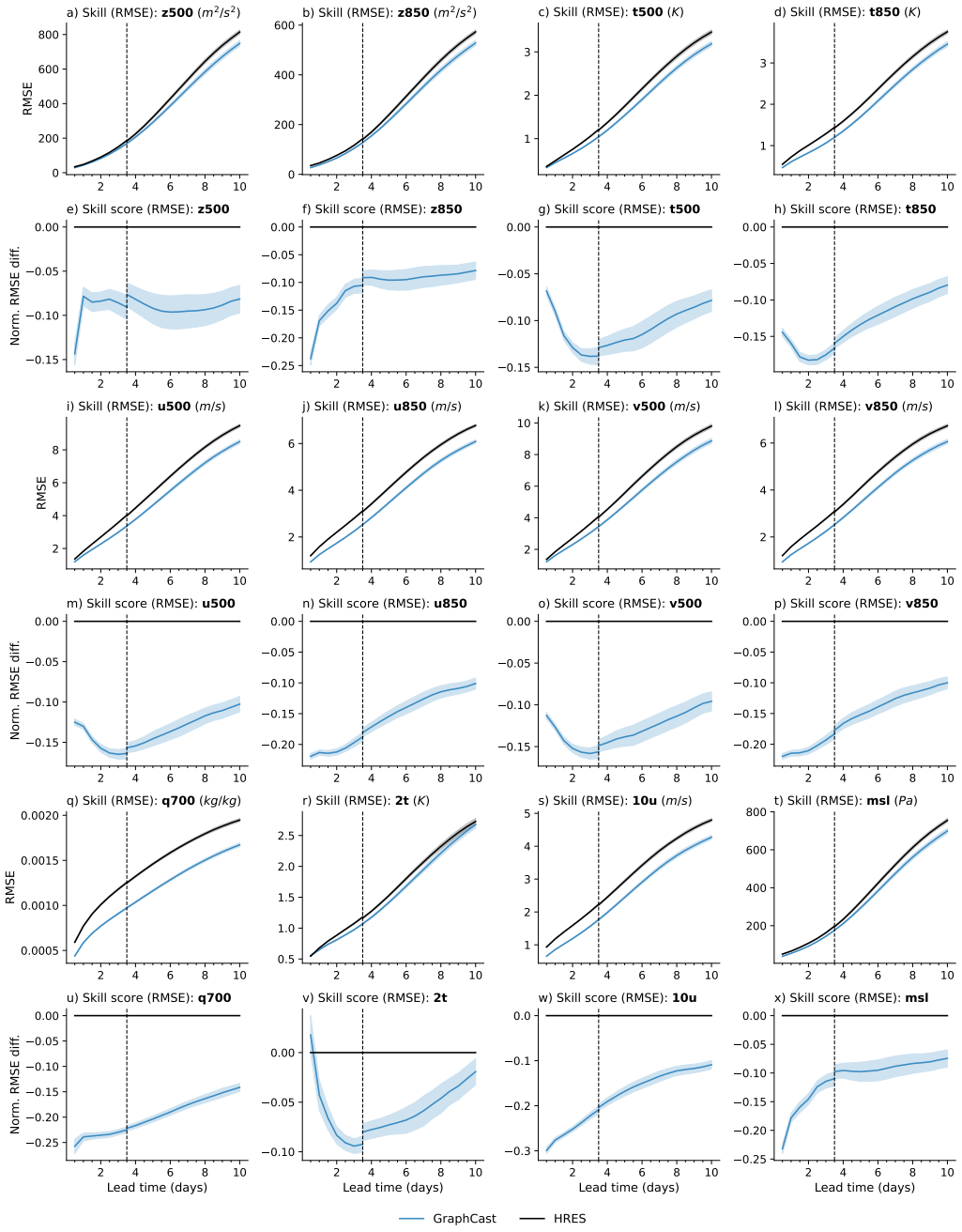

Fig. S10: **GraphCast's RMSE skill versus HRES in 2018** ***(lower is better)*** Rows 1, 3 and 5 show absolute RMSE for GraphCast (blue lines) and HRES (black lines), with 95% confidence interval error bars (see Section 5.4.3); rows 2, 4 and 6 show RMSE skill score (normalized RMSE differences between GraphCast's RMSE and HRES's) with 95% confidence interval error bars (see Section 5.4.4). Each subplot represents a single variable (and pressure level), as indicated in the subplot titles. The x-axis represents lead time, at 12-hour steps over 10 days. The y-axis represents (absolute or normalized) RMSE. The vertical dashed line represents 3.5 days, which which marks the transition from HRES forecasts initialized at 06z/18z, to forecast initialized at 00z/12z. This transition explains the discontinuity observed in GraphCast's skill score curves.

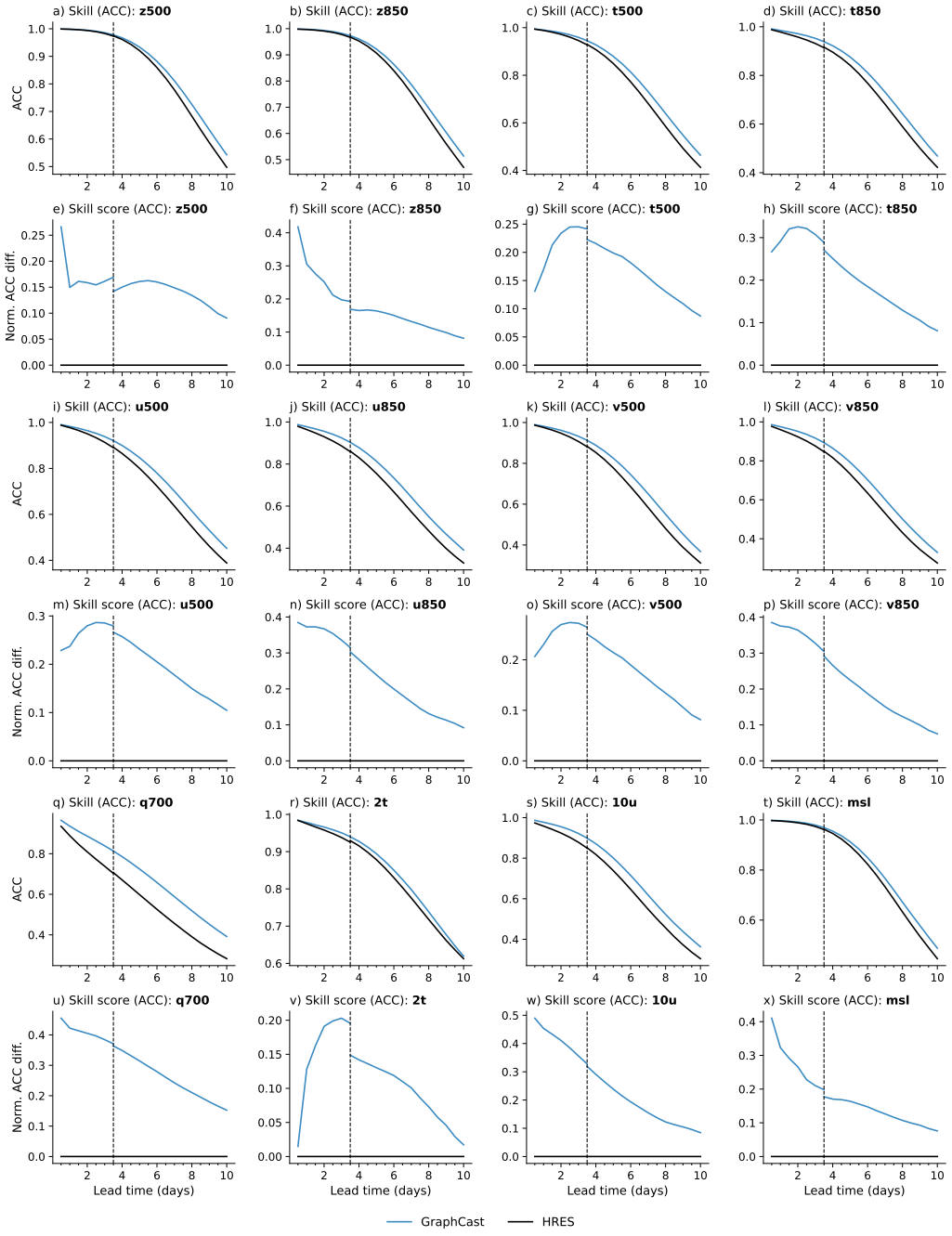

Fig. S11: **GraphCast's ACC skill versus HRES in 2018** ***(higher is better).*** Rows 1, 3 and 5 show absolute ACC for GraphCast (blue lines) and HRES (black lines); rows 2, 4 and 6 show ACC skill score (normalized ACC differences between GraphCast's RMSE and HRES's). Each subplot represents a single variable (and pressure level), as indicated in the subplot titles. The x-axis represents lead time, at 12-hour steps over 10 days. The y-axis represents (absolute or normalized) ACC. The vertical dashed line represents 3.5 days, which which marks the transition from HRES forecasts initialized at 06z/18z, to forecast initialized at 00z/12z. This transition explains the discontinuity observed in GraphCast's skill score curves.

Lead time	12 hours			2 days			10 days		
Variable	p	t	n_{eff}	p	t	n_{eff}	p	t	n_{eff}
Z50	$< 10^{-9}$	11	27.8	$< 10^{-9}$	6.87	37.3	$< 10^{-9}$	14.9	84.5
Z100	0.0045	−2.85	88	0.044	2.01	38.3	0.0088	−2.63	178
Z300	$< 10^{-9}$	−23.3	239	$< 10^{-9}$	−17.9	311	$< 10^{-9}$	−12.7	275
Z500	$< 10^{-9}$	−30.1	319	$< 10^{-9}$	−20.6	337	$< 10^{-9}$	−12.8	268
Z700	$< 10^{-9}$	−50	465	$< 10^{-9}$	−28.3	334	$< 10^{-9}$	−12.3	257
Z850	$< 10^{-9}$	−59.8	452	$< 10^{-9}$	−33.5	332	$< 10^{-9}$	−12.1	259
Z1000	$< 10^{-9}$	−76.9	462	$< 10^{-9}$	−40.5	349	$< 10^{-9}$	−12.1	261
T50	$< 10^{-9}$	31.8	27.2	$< 10^{-9}$	8.56	32.9	$< 10^{-9}$	40.5	54.3
T100	$< 10^{-9}$	20.6	94.5	0.32	−0.995	35.6	1.7×10^{-6}	4.83	55.2
T300	0.35	0.941	228	$< 10^{-9}$	−59.6	251	$< 10^{-9}$	−20	198
T500	$< 10^{-9}$	−40.7	224	$< 10^{-9}$	−71.2	366	$< 10^{-9}$	−17.1	279
T700	$< 10^{-9}$	−60.4	186	$< 10^{-9}$	−72.1	202	$< 10^{-9}$	−17.6	298
T850	$< 10^{-9}$	−78.4	243	$< 10^{-9}$	−90.3	229	$< 10^{-9}$	−16.9	244
T1000	$< 10^{-9}$	−23	82.9	$< 10^{-9}$	−59.7	169	$< 10^{-9}$	−11.8	275
U50	$< 10^{-9}$	25.5	20.9	$< 10^{-9}$	10.1	20.9	$< 10^{-9}$	19.6	55
U100	4.5×10^{-5}	4.1	158	$< 10^{-9}$	−14.7	86	$< 10^{-9}$	−10.1	133
U300	$< 10^{-9}$	−41.8	235	$< 10^{-9}$	−76.9	295	$< 10^{-9}$	−25.3	294
U500	$< 10^{-9}$	−114	339	$< 10^{-9}$	−103	337	$< 10^{-9}$	−26.8	269
U700	$< 10^{-9}$	−162	285	$< 10^{-9}$	−124	263	$< 10^{-9}$	−27	227
U850	$< 10^{-9}$	−183	275	$< 10^{-9}$	−134	336	$< 10^{-9}$	−27.6	243
U1000	$< 10^{-9}$	−155	183	$< 10^{-9}$	−134	383	$< 10^{-9}$	−26.6	231
V50	$< 10^{-9}$	35.5	31.8	$< 10^{-9}$	9.96	36.4	0.34	0.951	175
V100	0.023	2.28	175	$< 10^{-9}$	−14	77.5	$< 10^{-9}$	−16.5	234
V300	$< 10^{-9}$	−27.2	198	$< 10^{-9}$	−78.7	343	$< 10^{-9}$	−19	261
V500	$< 10^{-9}$	−101	331	$< 10^{-9}$	−96	365	$< 10^{-9}$	−21	256
V700	$< 10^{-9}$	−159	297	$< 10^{-9}$	−127	315	$< 10^{-9}$	−25.3	241
V850	$< 10^{-9}$	−181	272	$< 10^{-9}$	−129	335	$< 10^{-9}$	−25.8	260
V1000	$< 10^{-9}$	−211	345	$< 10^{-9}$	−130	367	$< 10^{-9}$	−25.5	275
Q50	$< 10^{-9}$	24.5	43.9	$< 10^{-9}$	20.7	34.1	$< 10^{-9}$	8.25	56.6
Q100	1.8×10^{-8}	−5.69	177	$< 10^{-9}$	−8.91	77.6	7.9×10^{-5}	−3.97	22.5
Q300	$< 10^{-9}$	−170	224	$< 10^{-9}$	−139	188	$< 10^{-9}$	−42.5	125
Q500	$< 10^{-9}$	−70.6	78.9	$< 10^{-9}$	−137	214	$< 10^{-9}$	−40.4	129
Q700	$< 10^{-9}$	−54.2	50	$< 10^{-9}$	−150	180	$< 10^{-9}$	−49.2	166
Q850	$< 10^{-9}$	−128	92.1	$< 10^{-9}$	−222	199	$< 10^{-9}$	−61.4	163
Q1000	$< 10^{-9}$	−85.6	89.3	$< 10^{-9}$	−128	140	$< 10^{-9}$	−28.8	215
2T	0.037	2.09	38.9	$< 10^{-9}$	−23.4	108	0.00075	−3.39	249
10U	$< 10^{-9}$	−175	143	$< 10^{-9}$	−156	370	$< 10^{-9}$	−29.9	239
10V	$< 10^{-9}$	−281	298	$< 10^{-9}$	−160	365	$< 10^{-9}$	−28.8	283
MSL	$< 10^{-9}$	−82.4	501	$< 10^{-9}$	−41.2	360	$< 10^{-9}$	−12	260

Table S4: **Detailed significance test results for the comparison of GraphCast and HRES RMSE**. We list the p-value, the test statistic t and the effective sample size n_{eff} for all variables at three key lead times, and a subset of 7 levels chosen to include all cases where $p > 0.05$ at these lead times. Nominal sample size $n \in \{729, 730\}$.

GraphCast:<2019, in particular, have lower skill scores against HRES at early lead times for the 2021 test evaluation. We note that for other variables, GraphCast:<2018 and GraphCast:<2019 tend to still outperform HRES. These results highlight a key feature of GraphCast, in allowing performance to be automatically improved by re-training on recent data.

7.1.4 Precipitation

Precipitation is an important variable to predict and GraphCast models its 6h-accumulation at every step. However, for data quality reasons and lack of fair evaluation for the baseline HRES, we left precipitation out of the scope of this work since the beginning of the project, and simply kept it as an auxiliary variable. For the two aforementioned reasons, expanded below, we do not make any claims about performance on precipitation and leave this important thread of research for future work. We provide the following results for completeness and illustrative purpose.

Because precipitation data in ERA5 is known to have significant biases (*23*), in this work, we did not focus on improving its prediction as GraphCast would learn to reproduce the same biases. We simply kept it as an auxiliary variable that could provide useful information to the model. Because of this, we did not treat precipitation differently than any other variable: it is normalized in the same way (centered by mean, and scaled by standard deviation), and trained with the same MSE loss. Both those approaches are appropriate for variables with distributions close to Gaussian, however precipitation is sparse and very non-Gaussian variable, which can lead to sub-optimal training. Future work should address these limitations to improve the statistical modeling of precipitation.

Since there is no precipitation analysis available in the operational ECMWF products (HRES-fc0 and HRES Analysis), we used ERA5 reanalysis as targets to compute metrics for HRES. This differs from the evaluation protocol used in the rest of the paper, and puts HRES at a disadvantage, as explained in Section 5.2.1. With this important caveat in mind, we evaluate

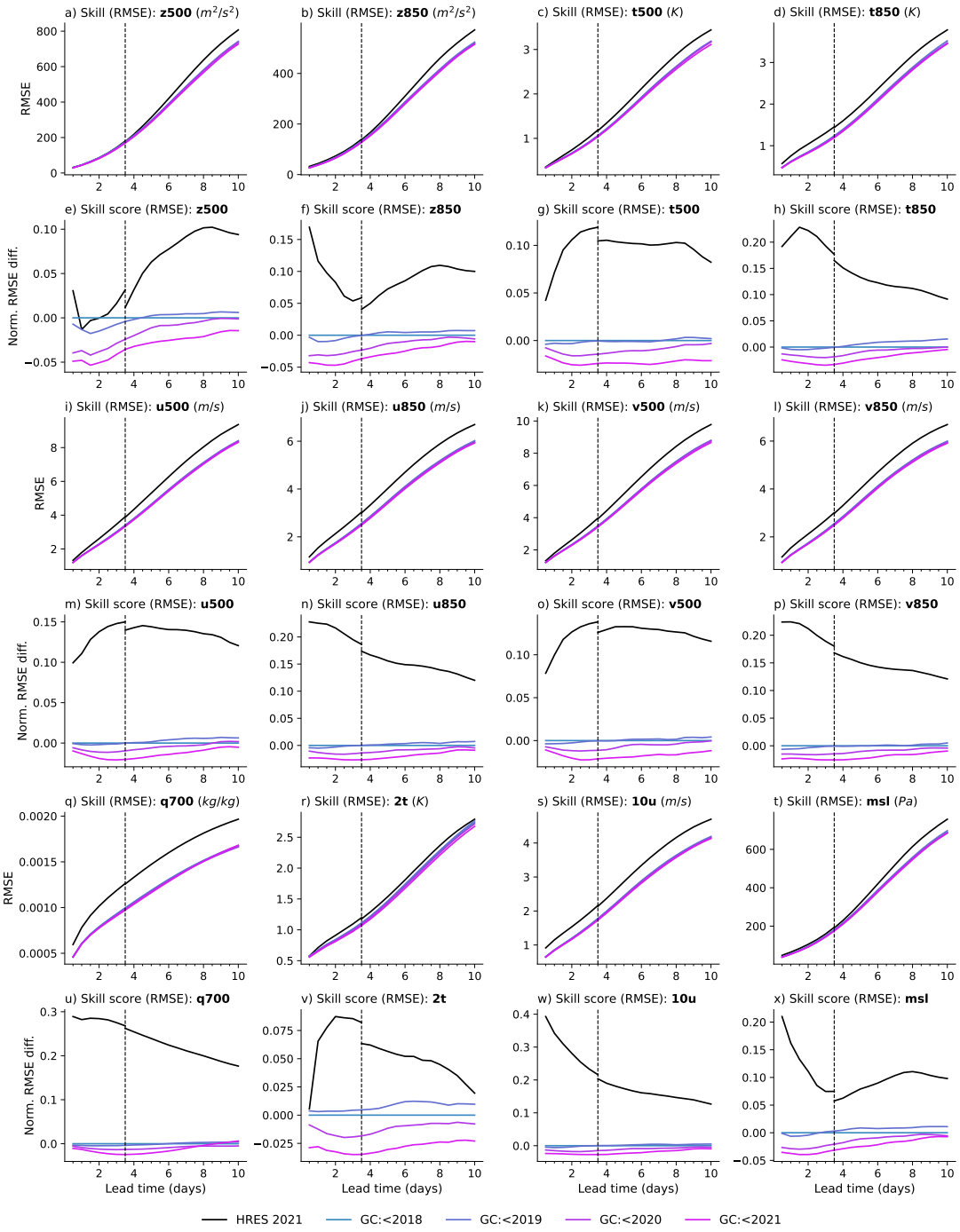

Fig. S12: **Effects of data recency.** Each color line in the plot represent a variant of GraphCast trained on different amounts of available data, while the black line represents HRES. All models are evaluated on the test year 2021. Rows 1, 3, and 5 show the RMSE, and rows 2, 4, and 6 show the normalized RMSE difference with respect to GraphCast:<2018. Each subplot represents a single variable (and pressure level), as indicated in the subplot titles. GraphCast's performance can be improved by retraining on the most recent data.

GraphCast and HRES both against ERA5 targets using RMSE and Stable Equitable Error in Probability Space (SEEPS) (*69–71*) of total precipitation over 6-hour and 24-hour intervals (see Figure S13). For the computation of SEEPS, we use the same equations, and the same parameters as in (*71*). We summarize this computation in the following lines. Predictions and targets are categorized as dry, light-rain, and heavy-rain over a given interval or "time window" (6 hour or 24 hour). Dry / light-rain / heavy-rain monthly climatology is computed at 0.25° resolution between 1980 and 2015. A "dry" time window occurs when there is less than 0.25 mm of precipitation over the interval. The threshold for light-rain / heavy-rain time-windows, is set location-wise, such that, in each location light-rain events occur twice as often as heavy-rain events. Regions for which less than 10% of the fraction of all windows are considered to be dry (very wet regions), and for which more than 85% of the fraction of all intervals are considered to be dry (very dry regions), are not included as part of the analysis. For each location and each time window, SEEPS is computed as,

$$\text{SEEPS} = \sum_{v,f} p_{v,f} s_{v,f} \tag{34}$$

where v is the actual category, f is the predicted category, and $p_{v,f}$ are the elements of the contingency matrix $P = \{p_{v,f}\}$ (which is 3x3 and, for a single deterministic prediction, contains all zeros except for a single 1 in the location that corresponds to the predicted category row and actual category column); and $s_{v,f}$ are the elements of the scoring matrix $S = \{s_{v,f}\}$,

$$S = \{s_{v,f}\} = \frac{1}{2} \begin{bmatrix} 0 & \frac{1}{1-p_1} & \frac{1}{p_3} + \frac{1}{1-p_1} \\ \frac{1}{p_1} & 0 & \frac{1}{p_3} \\ \frac{1}{p_1} + \frac{1}{1-p_3} & \frac{1}{1-p_3} & 0 \end{bmatrix} \tag{35}$$

such that p_1, and p_3 are the climatological probability of dry events, and heavy-rain events respectively, for that specific location. SEEPS is then averaged across all locations (with an area weighted averages), and across all initialization times.

The results indicate that total precipitation RMSE *(lower is better)* for GraphCast is about

30% better than HRES for 6-hour accumulations (Figure S13c), and 20% better than HRES for 24-hour accumulations (Figure S13d). Looking at 1-SEEPS, which is a more appropriate metrics for precipitation prediction skill *(higher is better)*, we observe however, that GraphCast is only better than HRES for 6-hour accumulations (Figure S13e), but not for 24-hour accumulations (Figure S13f). Furthermore, we noticed that using a threshold of 0.25 mm for dry regions over 6h periods (as in (*71*)), was leading to about 20% of the world being filtered out as "very dry" regions. To compensate for this we repeated the analysis using a threshold of 0.1 mm dry threshold (dashed line in the plots), which results in a similar fraction of "very dry" regions as 0.25 mm over 24h (about 10%). We noticed that while this change does not affect HRES performance much, it does makes the SEEPS performance of GraphCast worse. These results indicate that the metrics themselves seem to be sensitive to different biases in GraphCast and HRES, which makes them hard to interpret without further analysis (which is out of the scope of this work). We speculate we might be seeing two different trends. First, GraphCast may tend to underestimate precipitation compared to HRES, and this is why it worsens with a lower threshold parameter. It may be possible to correct for this via bias/variance correction, however it would be preferable to train the model with a training loss more appropriate for precipitation. Second, HRES may be better at predicting total precipitation volumes over a long intervals (e.g. 24h), but not very precise about when exactly it is going to happen. On the other hand, GraphCast may be better at predicting precise time intervals ('6h'), but may be worse at predicting the exact total volume.

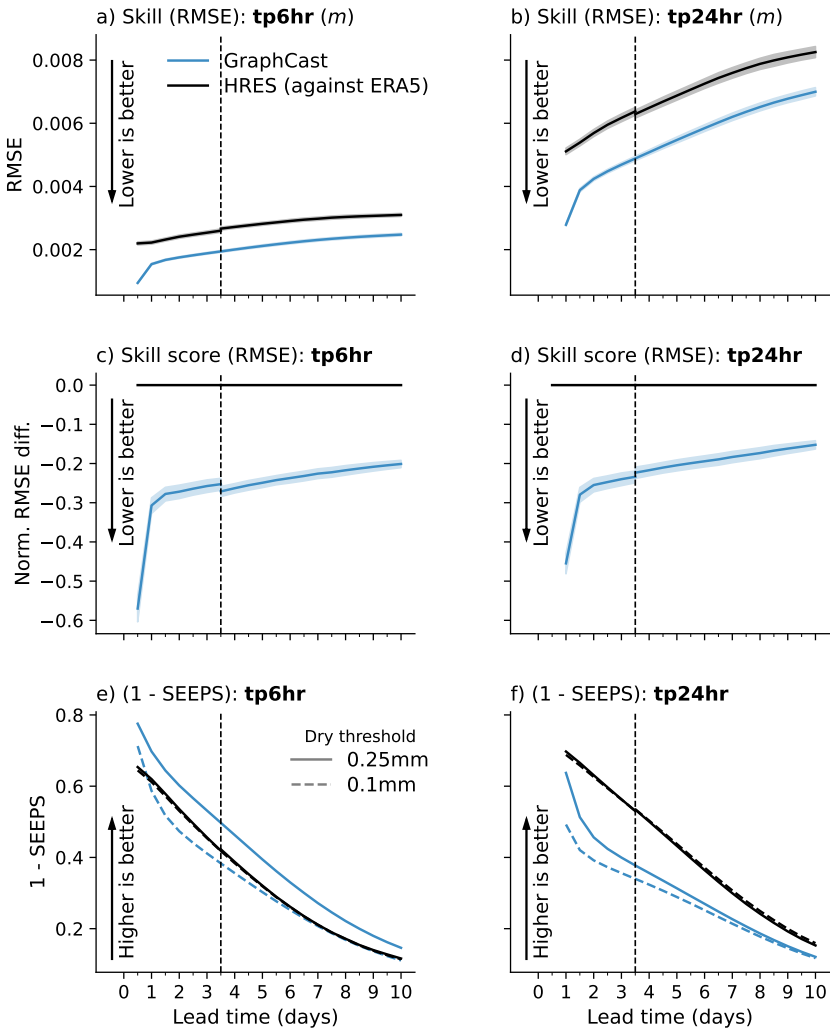

Fig. S13: **GraphCast's skill versus HRES in 2018 predicting 6 hour (a, c, e) and 24 hour (b, d, f) total precipitation** for GraphCast (blue lines) and HRES (black lines). (a,b) show absolute RMSE *(lower is better)* with 95% confidence interval error bars (see Section 5.4.3); (c,d) RMSE skill score *(lower is better)* (normalized RMSE differences between GraphCast's RMSE and HRES's) with 95% confidence interval error bars (see Section 5.4.4); and (e,f) 1-SEEPS *(higher is better)* (*69–71*), which is shown for two different dry thresholds: 0.25 mm of rain per 6h/24h interval (solid line), and 0.1 mm of rain per 6h/24h interval (dashed line). Note, in this case since there is no precipitation analysis available in the operational ECMWF products (HRES-fc0 and HRES Analysis), we used ERA5 reanalysis as targets to compute metrics for HRES. The x-axis represents lead time, at 12-hour steps over 10 days. The y-axis represents (absolute or normalized) RMSE. The vertical dashed line represents 3.5 days, which which marks the transition from HRES forecasts initialized at 06z/18z, to forecast initialized at 00z/12z. This transition explains the discontinuity observed in GraphCast's skill score curves.

7.2 Disaggregated results

7.2.1 RMSE by region

Per-region evaluation of forecast skill is provided in Figures S15 and S16, using the same regions and naming convention as in the ECMWF scorecards (https://sites.ecmwf.int/ifs/scorecards/scorecards-47r3HRES.html). We added some additional regions for better coverage of the entire planet. These regions are shown in Figure S14.

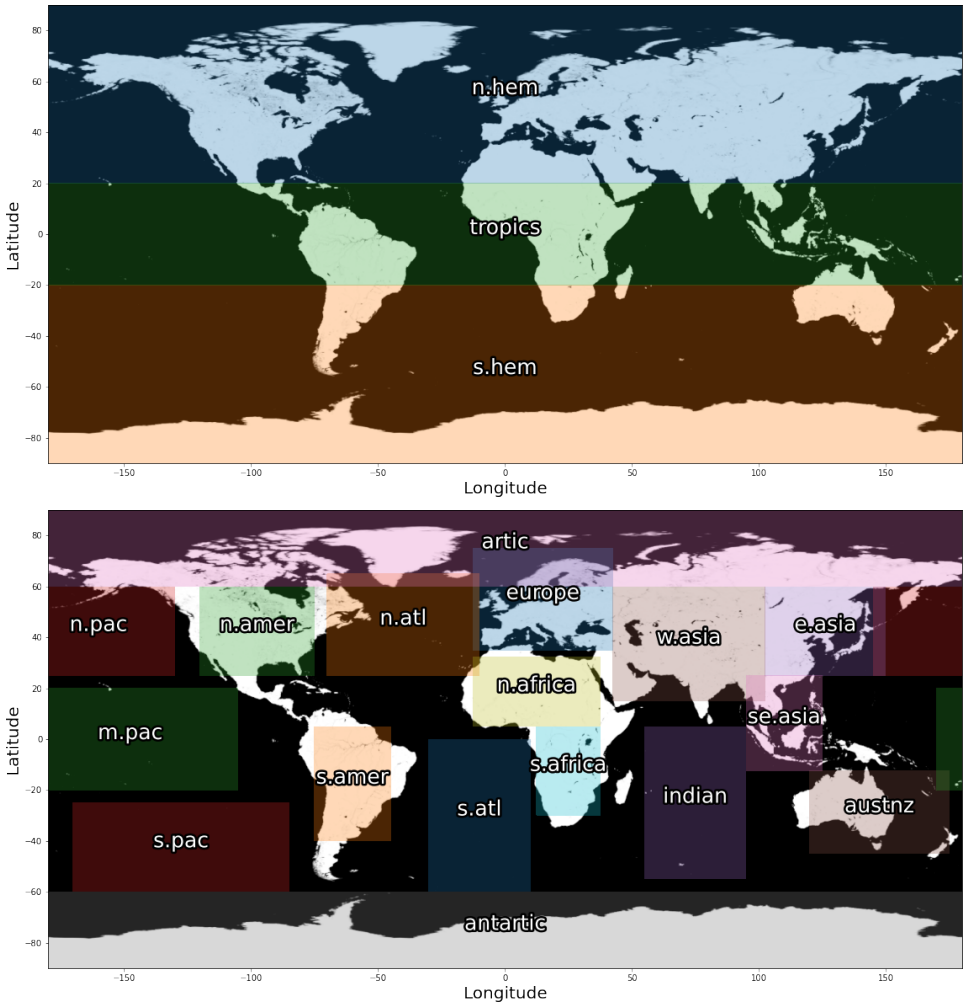

Fig. S14: **Region specification for the regional analysis.** We use the same regions and naming convention as in the ECMWF scorecards (https://sites.ecmwf.int/ifs/scorecards/scorecards-47r3HRES.html), and add some additional regions for better coverage of the entire planet. Per-region evaluation is provided in Figure S15 and Figure S16.

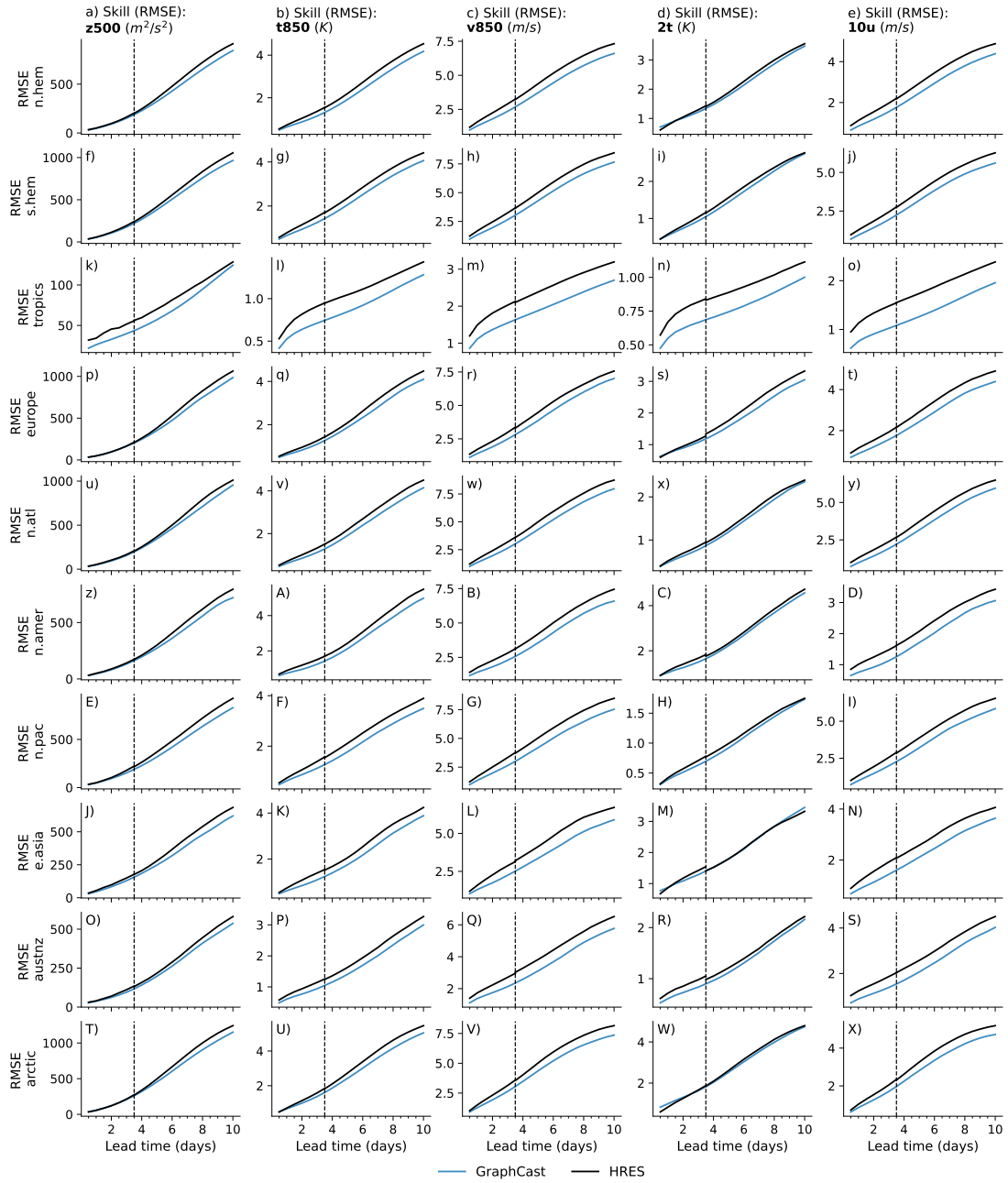

Fig. S15: **Skill (RMSE) of GraphCast versus HRES, per-region.** Each column is a different variable (and level), for a representative set of variables. Each row is a different region. The x-axis is lead time, in days. The y-axis is RMSE, with units specified in the column titles. GraphCast's RMSEs are the blue lines, and HRES's RMSEs are the black lines. The regions are: `n.hem`, `s.hem`, `tropics`, `europe`, `n.atl`, `n.amer`, `n.pac`, `e.asia`, `austnz`, and `arctic`. See Figure S14 for a legend of the region names.

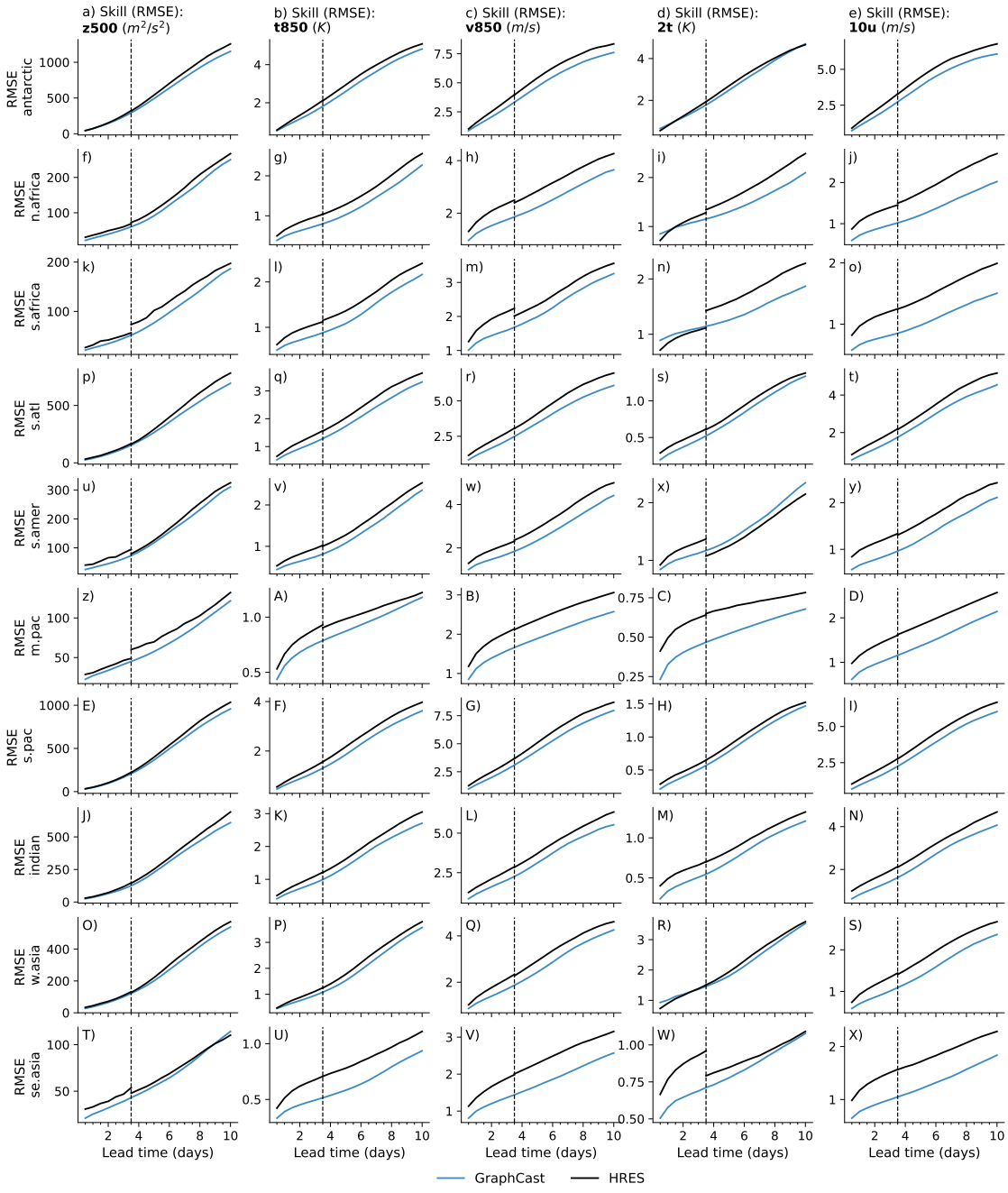

Fig. S16: **Skill (RMSE) of GraphCast versus HRES, per-region.** This plot is the same as Figure S15, except for regions (in Figure S14): `antarctic`, `n.africa`, `s.africa`, `s.atl`, `s.amer`, `m.pac`, `s.pac`, `indian`, `w.asia`, and `se.asia`.

7.2.2 RMSE skill score by latitude and pressure level

In Figure S17, we plot normalized RMSE differences between GraphCast and HRES, as a function of both pressure level and latitude. We plot only the 13 pressure levels from WeatherBench (*8*) on which we have evaluated HRES.

On these plots, we indicate at each latitude the mean pressure of the tropopause, which separates the troposphere from the stratosphere. We use values computed for the ERA5 dataset (1979-1993), given in Figure 1 of (*72*). These will not be quite the same as for ERA5 but are intended only as a rough aid to interpretation. We can see from the scorecard in Figure 2 that GraphCast performs worse than HRES at the lowest pressure levels evaluated (50hPa). Figure S17 shows that the pressure level at which GraphCast starts to get worse is often latitude-dependent too, in some cases roughly following the mean level of the tropopause.

The reasons for GraphCast's reduced skill in the stratosphere are currently poorly understood. We use a lower loss weighting for lower pressure levels and this may be playing some role (see comment on alternative weighting in Section 4.2); it is also possible that there may be differences between the ERA5 and HRES-fc0 datasets in the predictability of variables in the stratosphere.

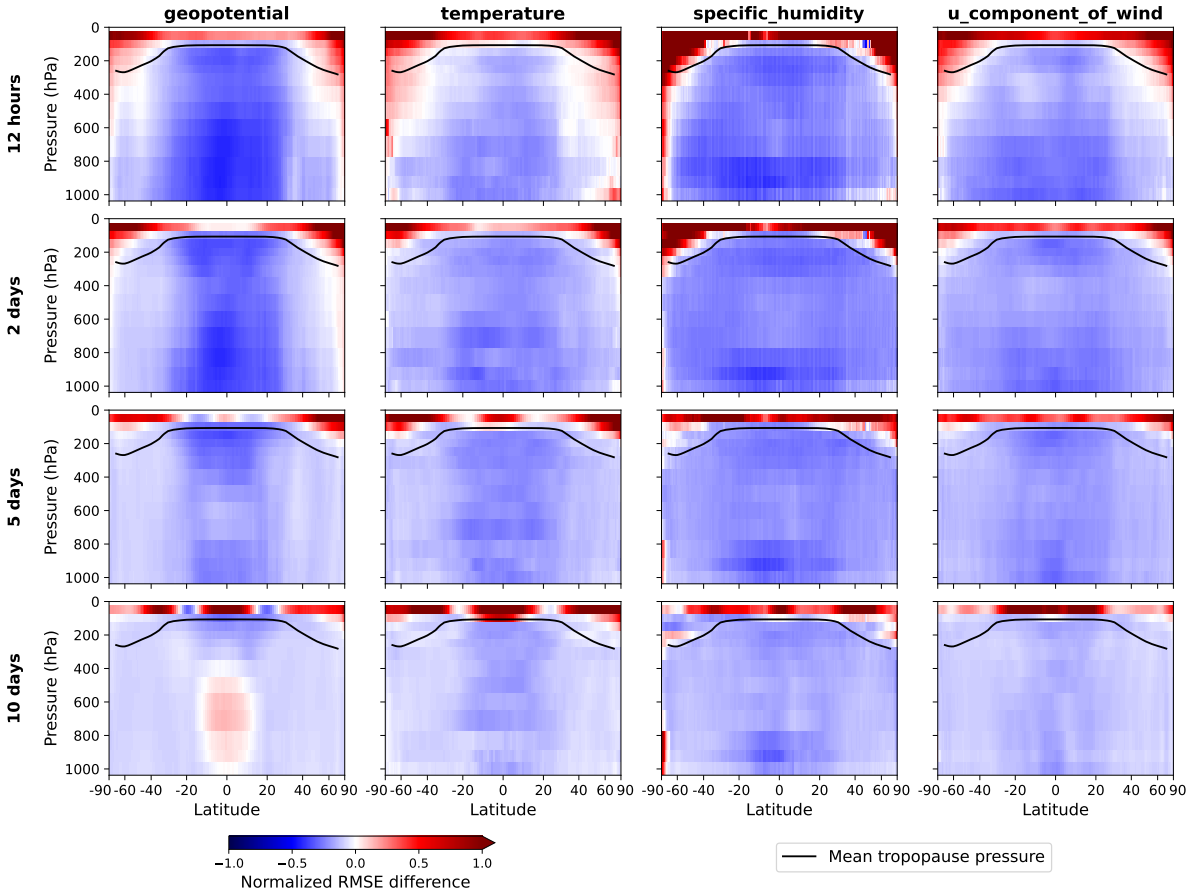

Fig. S17: **Normalized RMSE difference of Graphcast relative to HRES, by pressure level and latitude**. Black lines indicate the mean pressure of the tropopause at each latitude; the area above these lines in each plot corresponds roughly to the stratosphere. Latitude spacing is proportional to surface area. Red indicates that HRES has a lower RMSE than GraphCast, blue the opposite. GraphCast was evaluated using 06z/18z initializations; HRES was evaluated using 06z/18z initializations at 12 hour and 2 day lead times, and 00z/12z at 5 and 10 day lead times (see Section 5).

7.2.3 Biases by latitude and longitude

In Figures S18 to S23, we plot the mean bias error (MBE, or just 'bias', defined in Equation (26)) of GraphCast and HRES as a function of latitude and longitude, at three lead times: 12 hours, 2 days and 10 days. MBE is computed over the 2018 test set using 06z/18z initialization times, except in the case of HRES at 10 day lead time where only 00z/12z initializations are available.

In the plots for variables given on pressure levels, we have masked out regions whose surface elevation is high enough that the pressure level is below ground on average. We determine this to be the case when the surface geopotential exceeds a climatological mean geopotential at the same location and pressure level. In these regions the variable will typically have been interpolated below ground and will not represent a true atmospheric value.

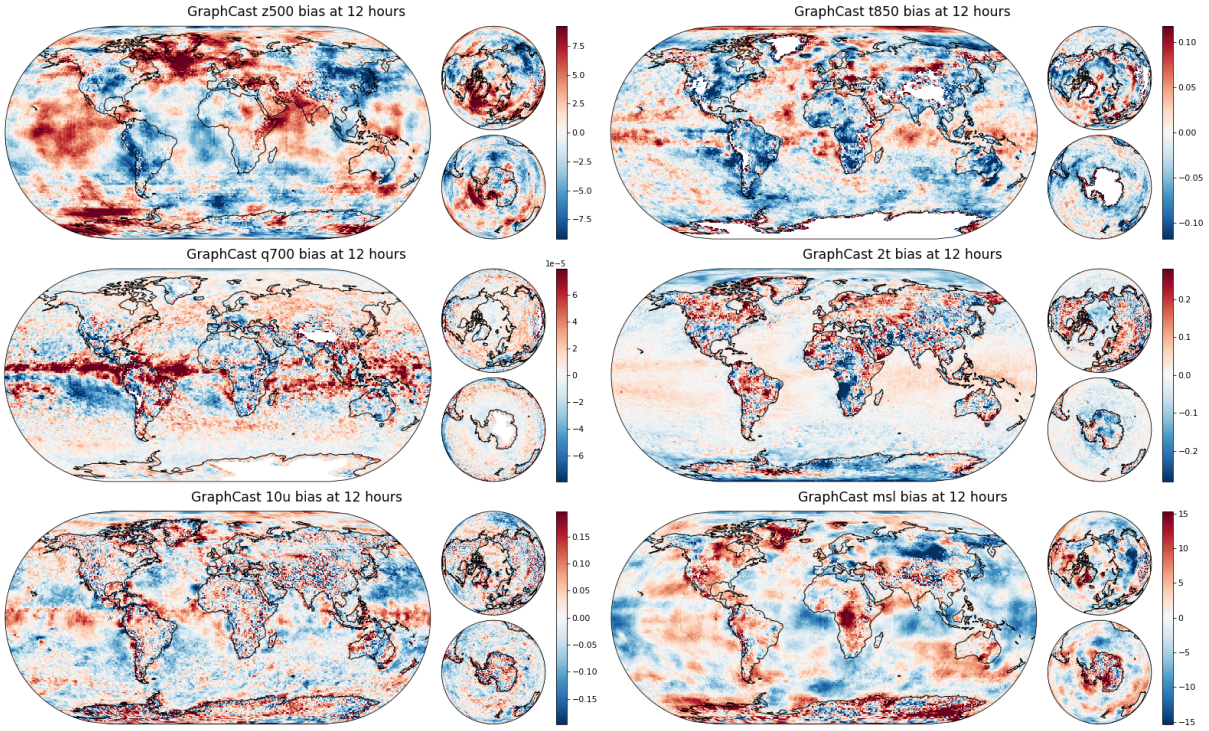

Fig. S18: **Mean bias error for GraphCast at 12 hour lead time**, over the 2018 test set at 06z/18z initializations

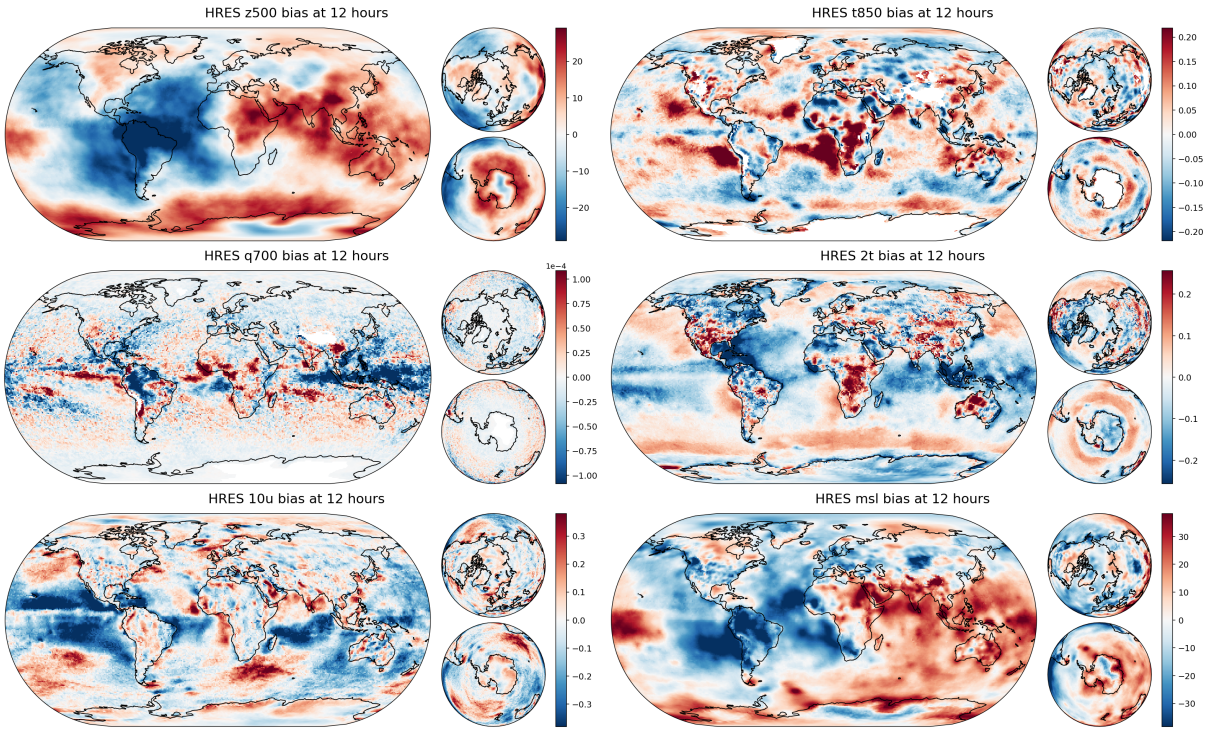

Fig. S19: **Mean bias error for HRES at 12 hour lead time**, over the 2018 test set at 06z/18z initializations.

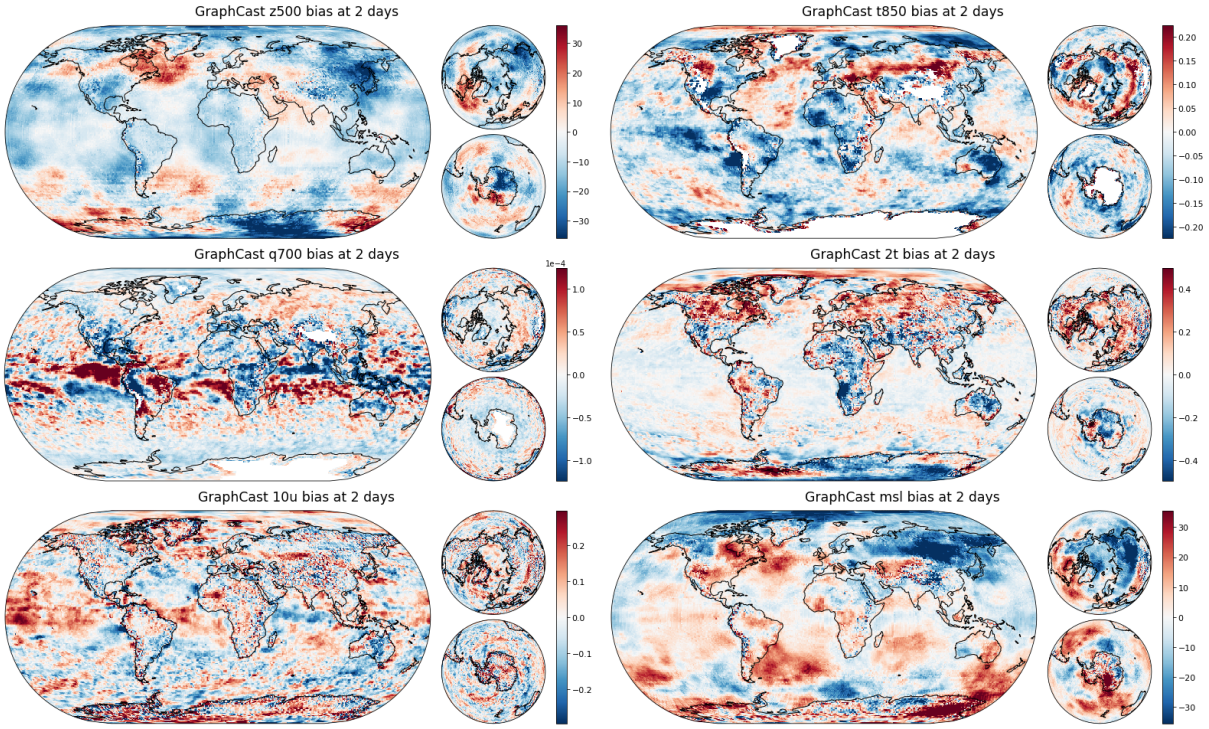

Fig. S20: **Mean bias error for GraphCast at 2 day lead time**, over the 2018 test set at 06z/18z initializations.

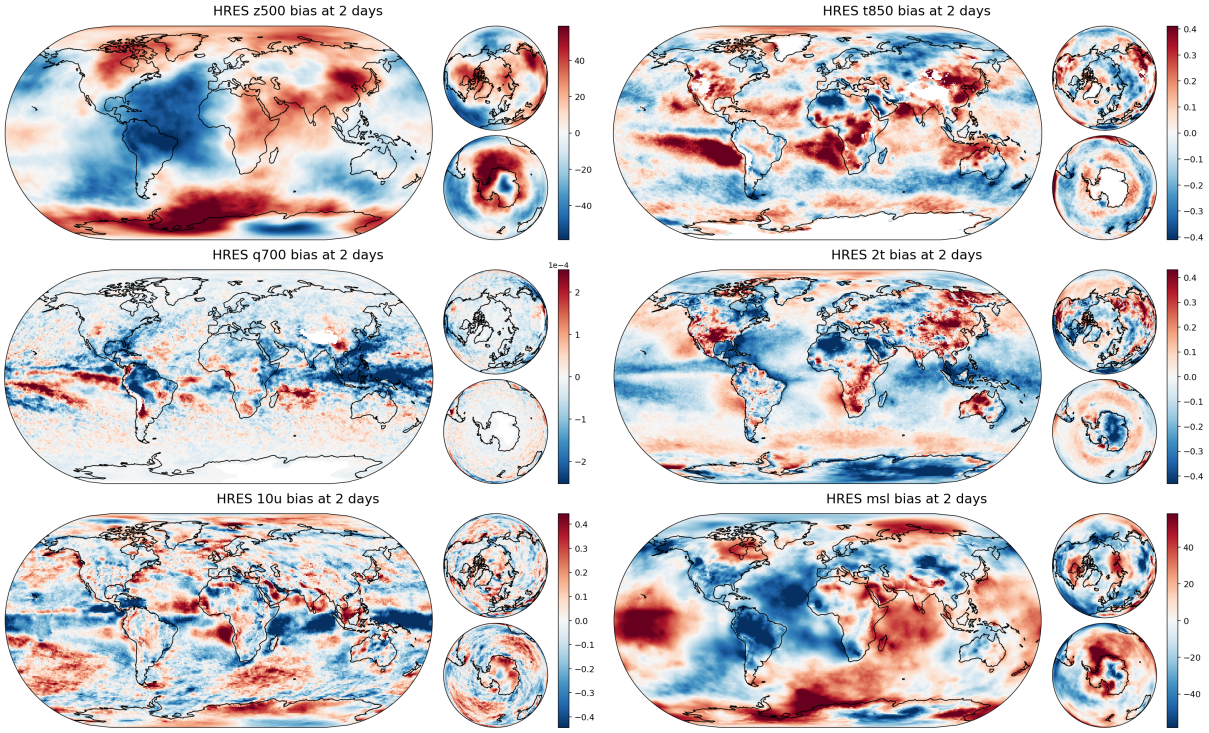

Fig. S21: **Mean bias error for HRES at 2 day lead time**, over the 2018 test set at 06z/18z initializations.

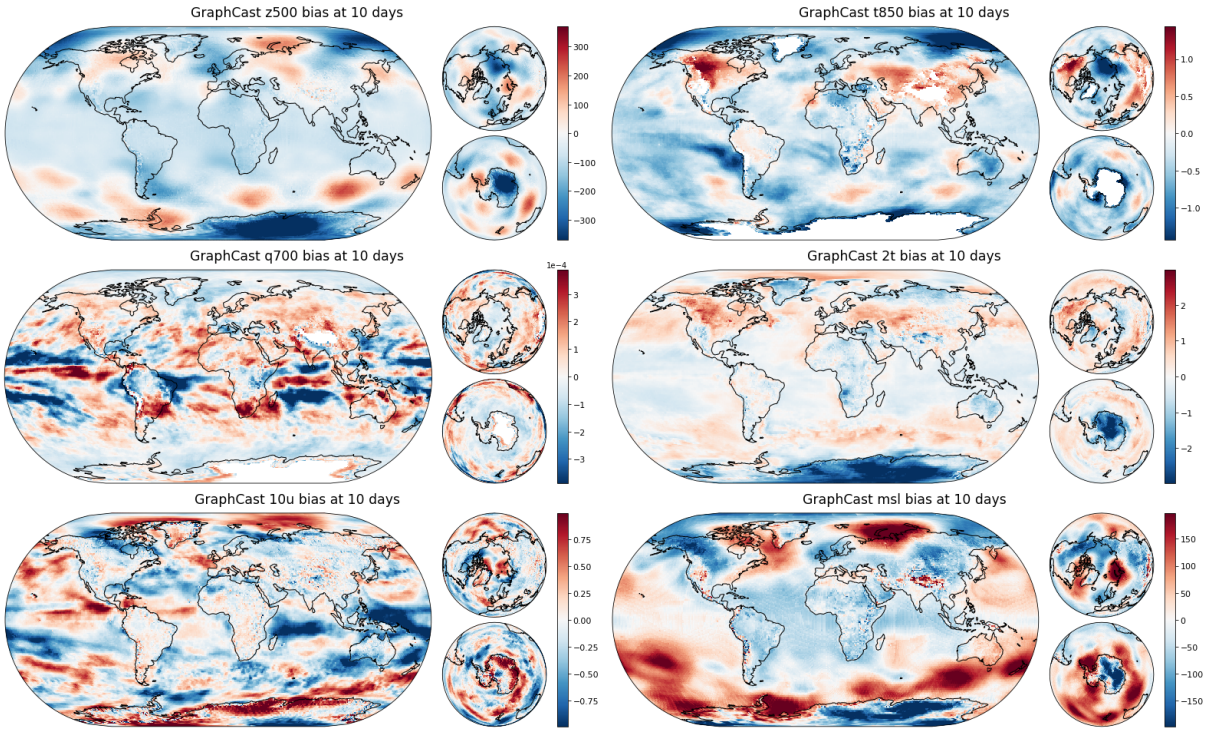

Fig. S22: **Mean bias error for GraphCast at 10 day lead time**, over the 2018 test set at 06z/18z initializations.

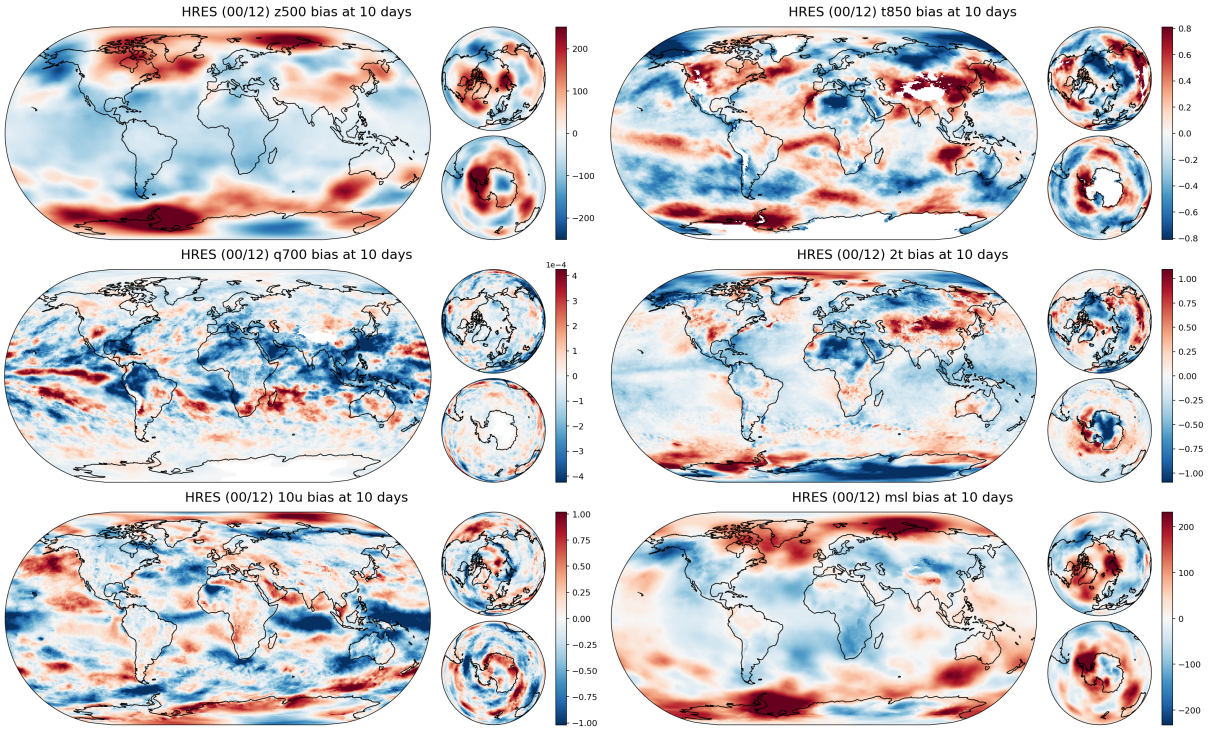

Fig. S23: **Mean bias error for HRES at 10 day lead time**, over the 2018 test set at 00z/12z initializations.

To quantify the average magnitude of the per-location biases shown in Figures S18 to S23, we computed the root-mean-square of per-location mean bias errors (RMS-MBE, defined in Equation (27)). These are plotted in Figure S24 for GraphCast and HRES as a function of lead time. We can see that GraphCast's biases are smaller on average than HRES' for most variables up to 6 days. However they generally start to exceed HRES' biases at longer lead times, and at 4 days in the case of 2m temperature. Forecasts of longer than 10 days would likely have much larger biases. There are, however, existing post-processing techniques that could potentially address this bias.

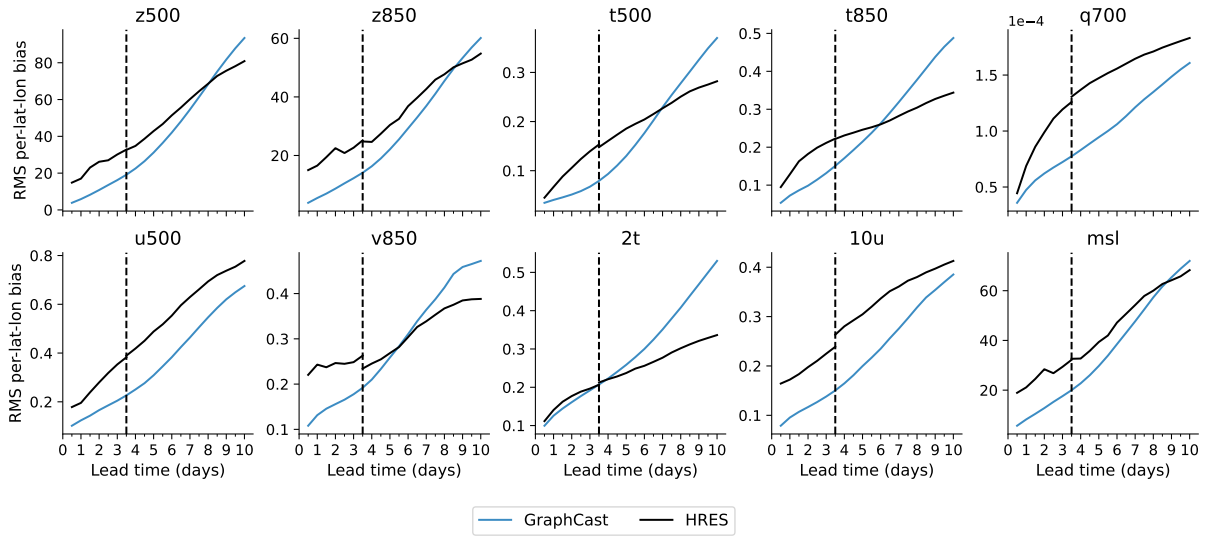

Fig. S24: **Root-mean-square of per-location biases for GraphCast and HRES** as a function of lead time.

7.2.4 RMSE skill score by latitude and longitude

In Figures S25 to S27, we plot the normalized RMSE difference between GraphCast and HRES by latitude and longitude. As in Section 7.2.3, for variables given on pressure levels, we have masked out regions whose surface elevation is high enough that the pressure level is below ground on average.

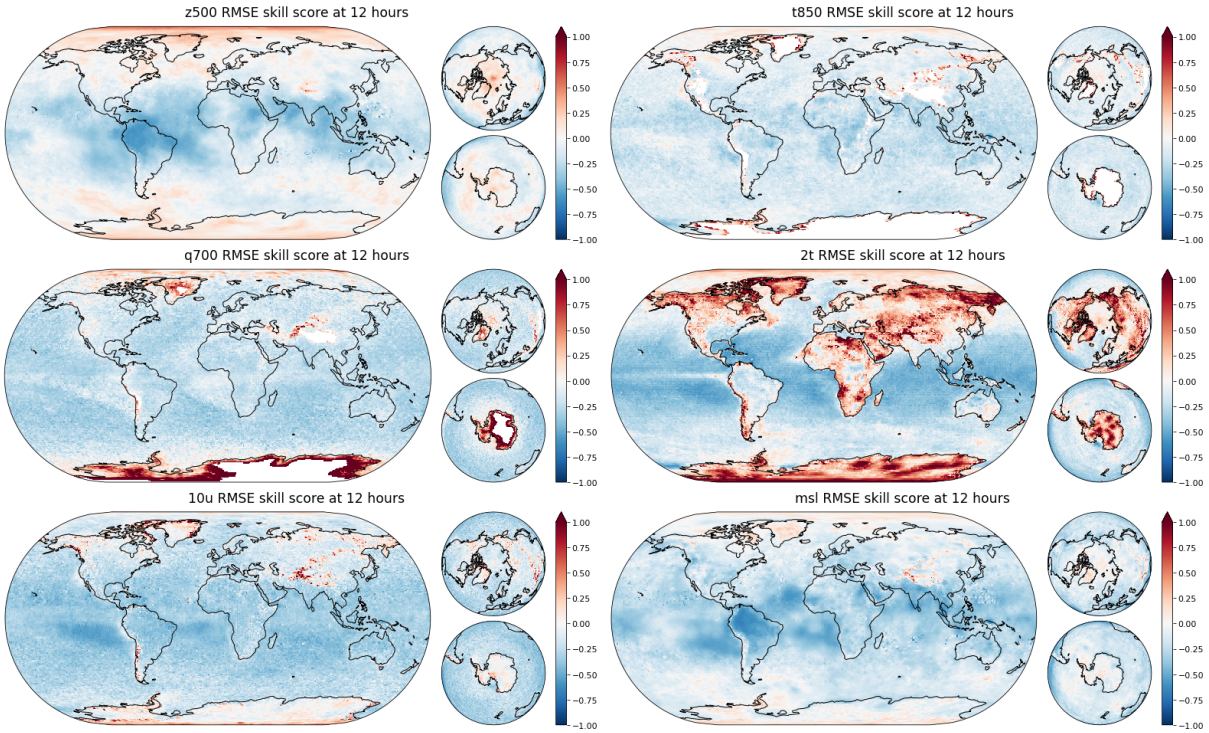

Fig. S25: **Normalized RMSE difference of GraphCast relative to HRES, by location, at 12 hours.** Blue indicates that GraphCast has greater skill than HRES, Red that HRES has greater skill.

Notable areas where HRES outperforms GraphCast include specific humidity near the poles (particularly the south pole); geopotential near the poles; 2m temperature near the poles and over many land areas; and a number of surface or near-surface variables in regions of high surface elevation (see also Section 7.2.5). GraphCast's skill in these areas generally improves over longer lead times. However HRES outperforms GraphCast on geopotential in some tropical regions at longer lead times.

At 12 hour and 2 day lead times both GraphCast and HRES are evaluated at 06z/18z initialization and validity times, however at 10 day lead times we must compare GraphCast at 06z/18z with HRES at 00z/12z (see Section 5). This difference in time-of-day may confound comparisons at specific locations for variables like 2m temperature (2T) with a strong diurnal cycle.

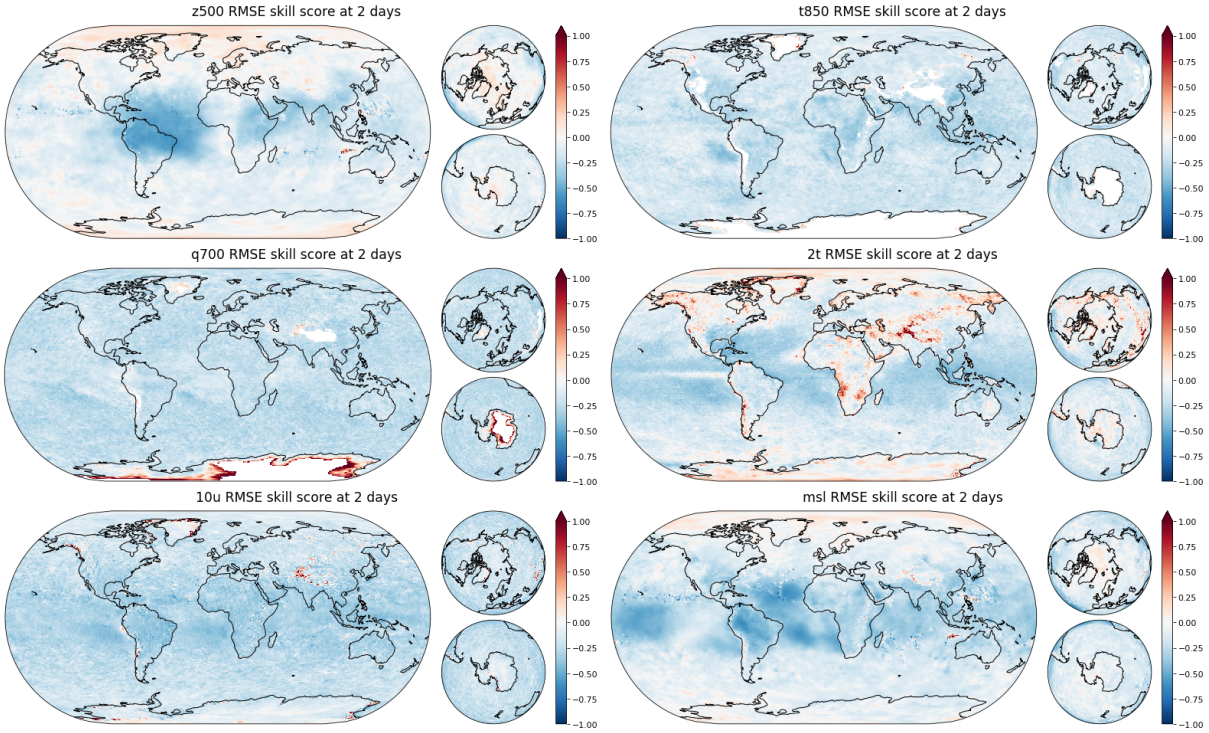

Fig. S26: **Normalized RMSE difference of GraphCast relative to HRES, by location, at 2 days.** Blue indicates that GraphCast has greater skill than HRES, Red that HRES has greater skill.

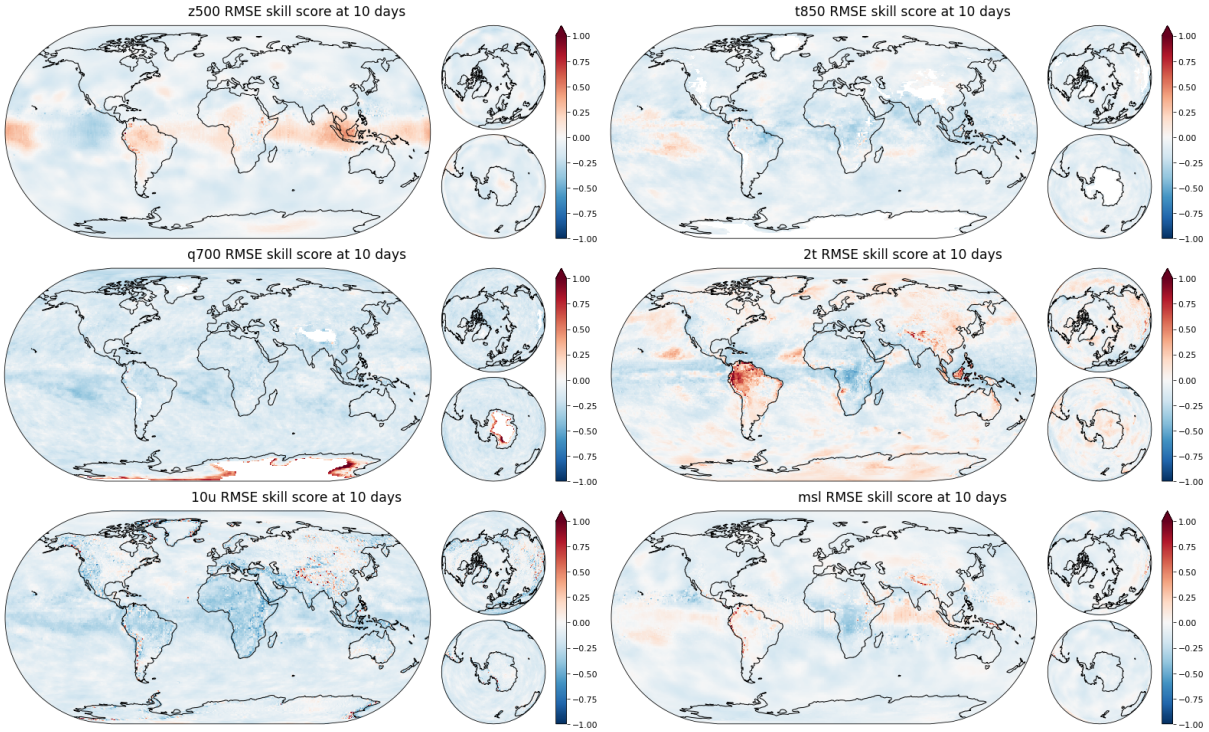

Fig. S27: **Normalized RMSE difference of GraphCast relative to HRES, by location, at 10 days.** Blue indicates that GraphCast has greater skill than HRES, Red that HRES has greater skill. In these 10 day plots we must compare GraphCast at 06z/18z with HRES at 00z/12z (see Section 5). This difference in time-of-day may confound some comparisons, e.g. of 2T.

7.2.5 RMSE skill score by surface elevation

In Figure S25, we can see that GraphCast appears to have reduced skill in high-elevation regions for many variables at 12 hour lead time. To investigate this further we divided the earth surface into 32 bins by surface elevation (given in terms of geopotential height) and computed RMSEs within each bin according to Equation (24). These are plotted in Figure S28.

At short lead times and especially at 6 hours, GraphCast's skill relative to HRES tends to decrease with higher surface elevation, in most cases dropping below the skill of HRES at sufficiently high elevations. At longer lead times of 5 to 10 days this effect is less noticeable, however.

We note that GraphCast is trained on variables defined using a mix of pressure-level coordinates (for atmospheric variables) and height above surface coordinates (for surface-level variables like 2m temperature or 10m wind). The relationship between these two coordinates systems depends on surface elevation. Despite GraphCast conditioning on surface elevation we conjecture that it may struggle to learn this relationship, and to extrapolate it well to the highest surface elevations. In further work we would propose to try training the model on a subset of ERA5's native model levels instead of pressure levels; these use a hybrid coordinate system (*73*) which follows the land surface at the lowest levels, and this may make the relationship between surface and atmospheric variables easier to learn, especially at high surface elevations.

Variables using pressure-level coordinates are interpolated below ground when the pressure level exceeds surface pressure. GraphCast is not given any explicit indication that this has happened and this may add to the challenge of learning to forecast at high surface elevations. In further work using pressure-level coordinates we propose to provide additional signal to the model indicating when this has happened.

Finally, our loss weighting is lower for atmospheric variables at lower pressure levels, and this may affect skill at higher-elevation locations. Future work might consider taking surface

elevation into account in this weighting.

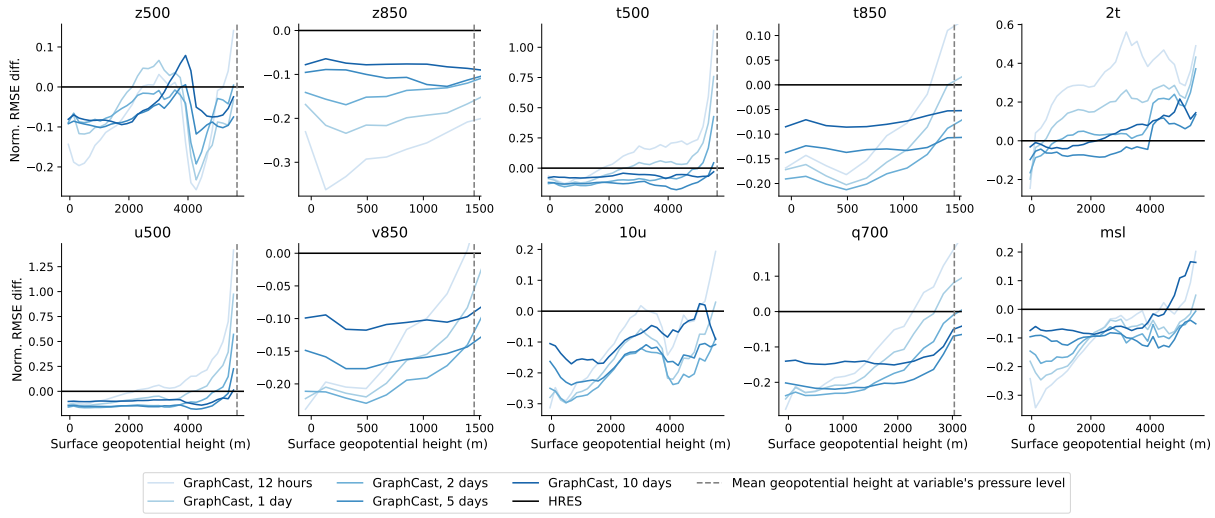

Fig. S28: **Normalized RMSE difference of GraphCast relative to HRES, by surface geopotential height.** For pressure-level variables, we crop the x-axis to exclude surface geopotential heights at which the variable is typically below ground (those greater than the mean geopotential height at the variable's pressure level, indicated via a dotted vertical line).

7.3 GraphCast ablations

7.3.1 Multi-mesh ablation

To better understand how the multi-mesh representation affects the performance of GraphCast, we compare GraphCast performance to a version of the model trained without the multi-mesh representation. The architecture of the latter model is identical to GraphCast (including same encoder and decoder, and the same number of nodes), except that in the process block, the graph only contains the edges from the finest icosahedron mesh M^6 (245,760 edges, instead of 327,660 for GraphCast). As a result, the ablated model can only propagate information with short-range edges, while GraphCast contains additional long-range edges.

Figure S29 (left panel) shows the scorecard comparing GraphCast to the ablated model. GraphCast benefits from the multi-mesh structure for all predicted variables, except for lead

times beyond 5 days at 50 hPa. The improvement is especially pronounced for geopotential across all pressure levels and for mean sea-level pressure for lead times under 5 days. The middle panel shows the scorecard comparing the ablated model to HRES, while the right panel compares GraphCast to HRES, demonstrating that the multi-mesh is essential for GraphCast to outperform HRES on geopotential at lead times under 5 days.

7.3.2 Effect of autoregressive training

We analyzed the performance of variants of GraphCast that were trained with fewer autoregressive (AR) steps[12], which should encourage them to improve their short lead time performance at the expense of longer lead time performance. As shown in Figure S30 (with the lighter blue lines corresponding to training with fewer AR steps) we found that models trained with fewer AR steps tended to trade longer for shorter lead time accuracy. These results suggest potential for combining multiple models with varying numbers of AR steps, e.g., for short, medium and long lead times, to capitalize on their respective advantages across the entire forecast horizon. The connection between number of autoregressive steps and blurring is discussed in Supplements Section 7.4.4.

[12]Each of these models were trained using a curriculum where the 1 AR-step model was fine-tuned for 1000 gradient updates each, on increasing numbers of AR steps, from 2-12 (see Section 4 and Figure S3 for details). Each model shown in Figure S30 completed its respective number of AR-step training. This means the higher AR-step models had slightly more training than the others, though we do not believe that each had generally converged, so training the lower AR-step models longer likely would not have made much difference.

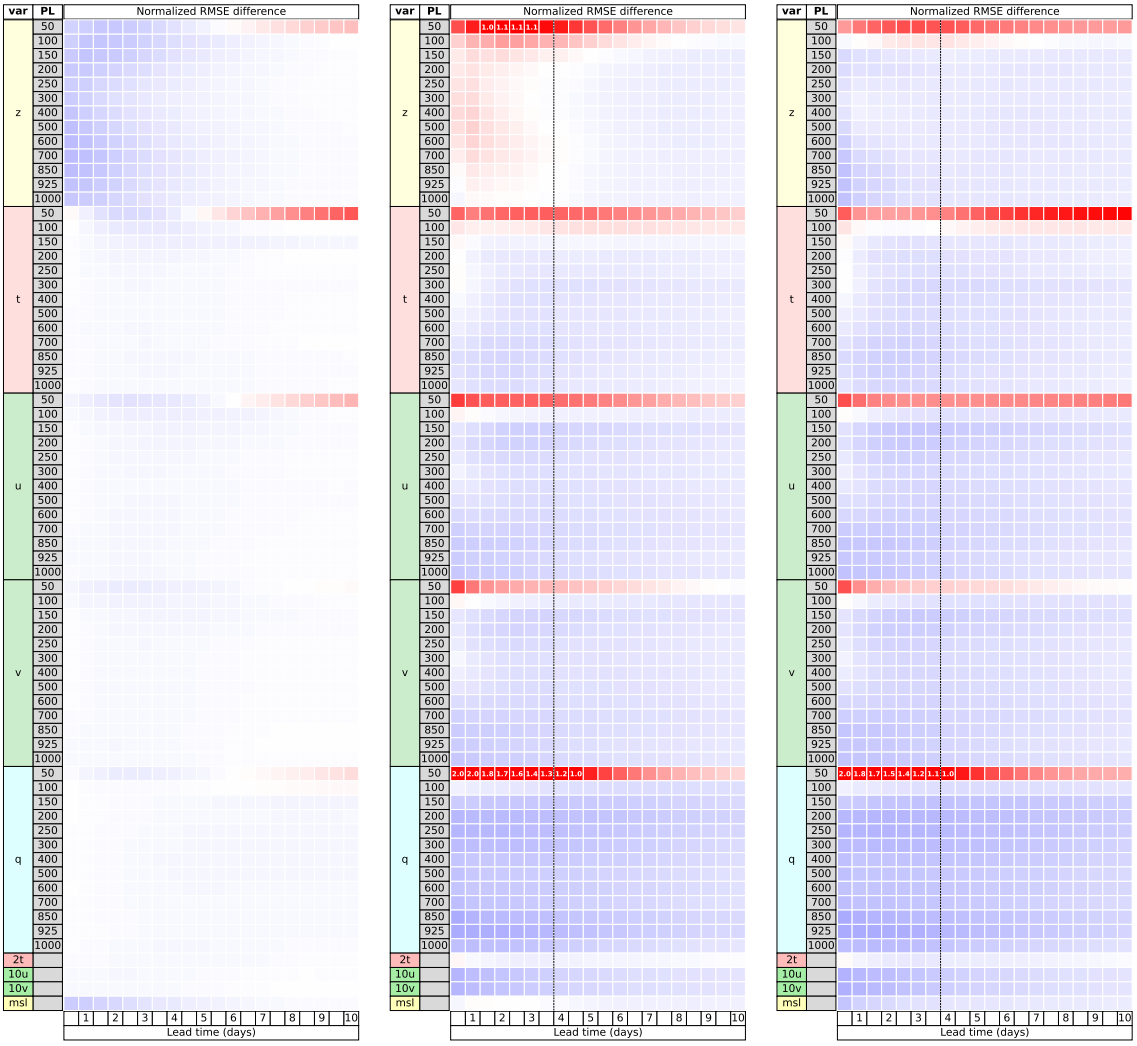

Fig. S29: **Scorecards comparing GraphCast to the ablated model without multi-mesh edges (left panel), the ablated model to HRES (middle panel) and GraphCast to HRES (right panel)**. In the left panel, blue cells represent variables and lead time where GraphCast is better than the ablated model, showing that training a model with the multi-mesh improves performance for all variables, except at 50 hPa past 5 days of lead time. In the middle panel, blue cells represent variables and lead time where the ablated model is better than HRES. Comparing the middle panel to the right one, where blue cells indicate that GraphCast is better than HRES, shows that the multi-mesh is necessary to outperform HRES on geopotential for lead times under 5 days.

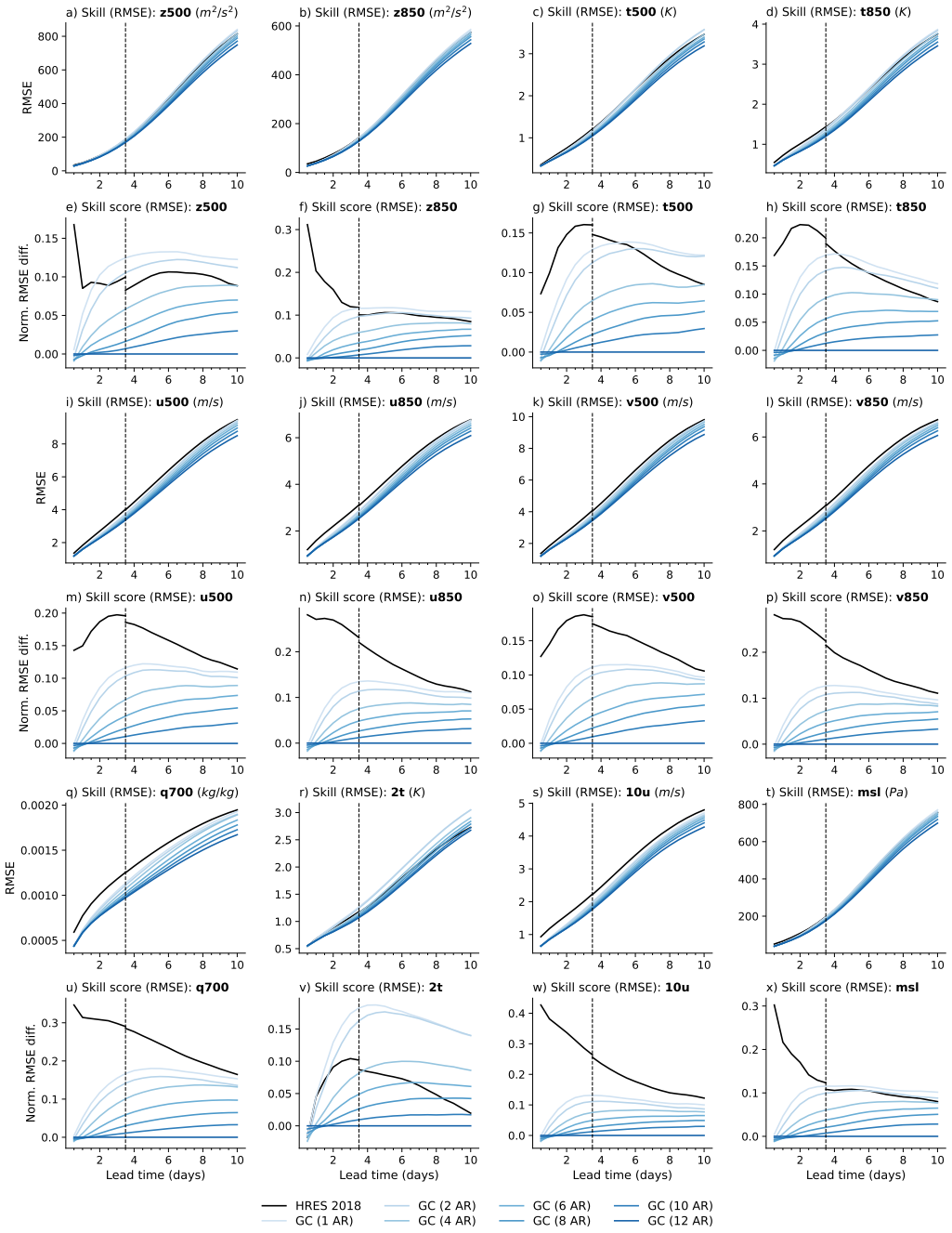

Fig. S30: **Effects of autoregressive training.** Each line in the plots represents GraphCast, fine-tuned with different numbers of autoregressive steps, where increasing numbers of steps are represented with darker shades of blue. Rows 1, 3 and 5 show absolute RMSE for GraphCast. Rows 2, 4 and 6 show normalized RMSE differences, with respect to our full 12 autoregressive-step (AR) GraphCast. Each subplot represents a single variable (and pressure level), as indicated in the subplot titles. The x-axis represents lead time, at 12-hour steps over 10 days. The y-axis represents (absolute or normalized) RMSE.

7.4 Optimized blurring

7.4.1 Effect on the comparison of skill between GraphCast and HRES

Generally GraphCast's MSE training objective encourages it to produce blurrier predictions at long lead times, which may not be desirable for some applications. It is also possible that GraphCast's RMSE skill advantage over HRES is due in part to the fact that blurred predictions can give lower RMSE under predictive uncertainty. To assess this possibility, we artificially blurred GraphCast's and HRES's forecasts by fitting RMSE-optimizing blurring filters independently for each model, and then compared their skills.

Figures S31 and S32 compares the resulting RMSE skills of HRES versus GraphCast before and after optimized blurring has been applied to both models. We can see that optimized blurring rarely changes the ranking of the two models, however it does generally narrow the gap between them.

7.4.2 Filtering methodology

We chose filters which minimize RMSE within the class of linear, homogeneous (location invariant), isotropic (direction invariant) filters on the sphere. These filters can be applied easily in the spherical harmonic domain, where they correspond to multiplicative filter weights that depend on the total wavenumber, but not the longitudinal wavenumber (*74*).

For each initialization d_0, lead time τ, variable and level j, we applied a discrete spherical harmonic transform (*62*) to predictions $\hat{x}_j^{d_0+\tau}$ and targets $x_j^{d_0+\tau}$, obtaining spherical harmonic coefficients $\hat{f}_{j,l,m}^{d_0+\tau}$ and $f_{j,l,m}^{d_0+\tau}$ for each pair of total wavenumber l and longitudinal wavenumber m. To resolve the 0.25° (28km) resolution of our grid at the equator, we use a triangular truncation at total wavenumber 719, which means that l ranges from 0 to $l_{max} = 719$, and for each l the value of m ranges from $-l$ to l.

We then multiplied each predicted coefficient $\hat{f}_{j,l,m}^{d_0+\tau}$ by a filter weight $b_{j,l}^{\tau}$, which is inde-

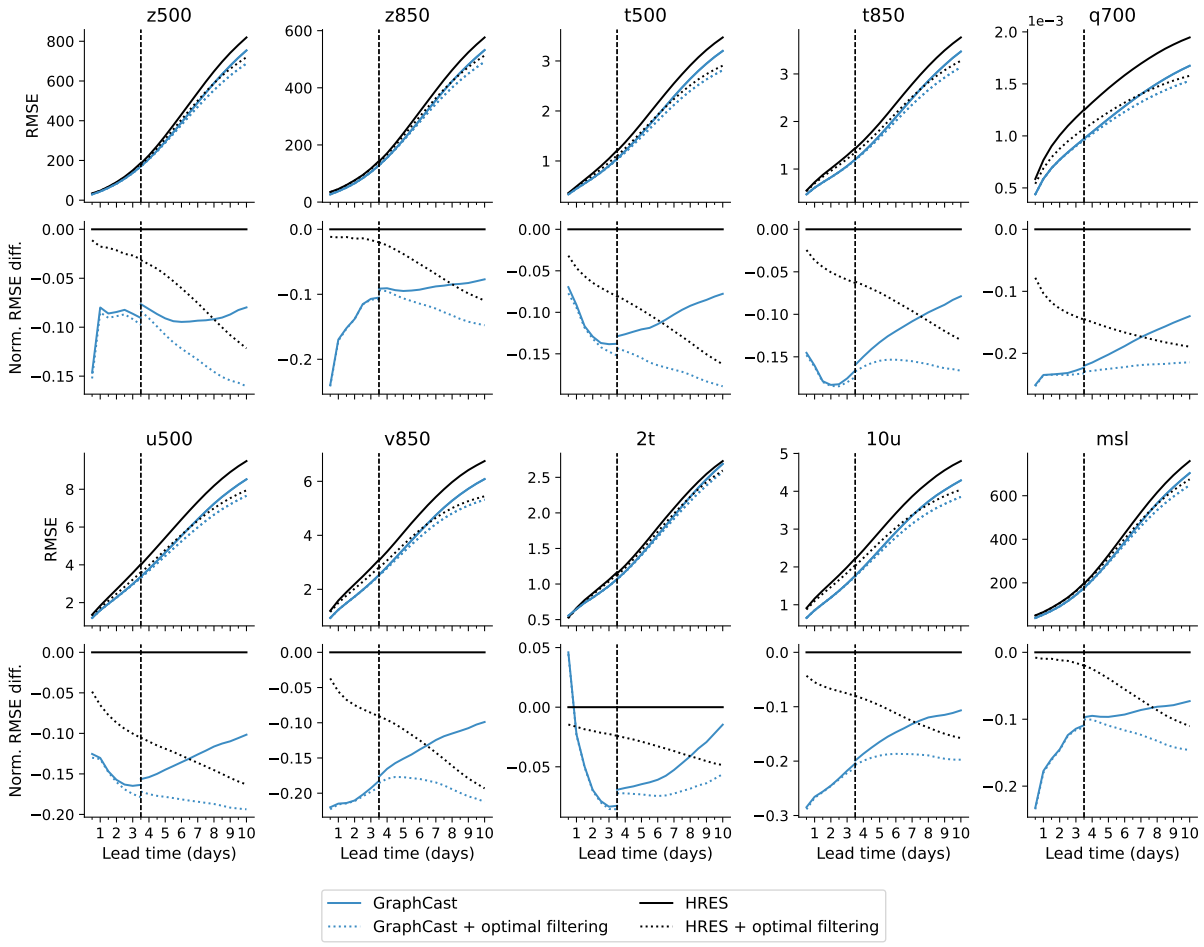

Fig. S31: **Effect of optimized blurring on GraphCast and HRES RMSE skill**. We show RMSEs for unfiltered predictions (solid lines) and optimally filtered predictions (dotted lines) for both GraphCast and HRES. Rows 1 and 3 show RMSEs, rows 2 and 4 show RMSE skill scores relative to unfiltered HRES. RMSEs are computed in the spherical harmonic domain (see Equation (22)).

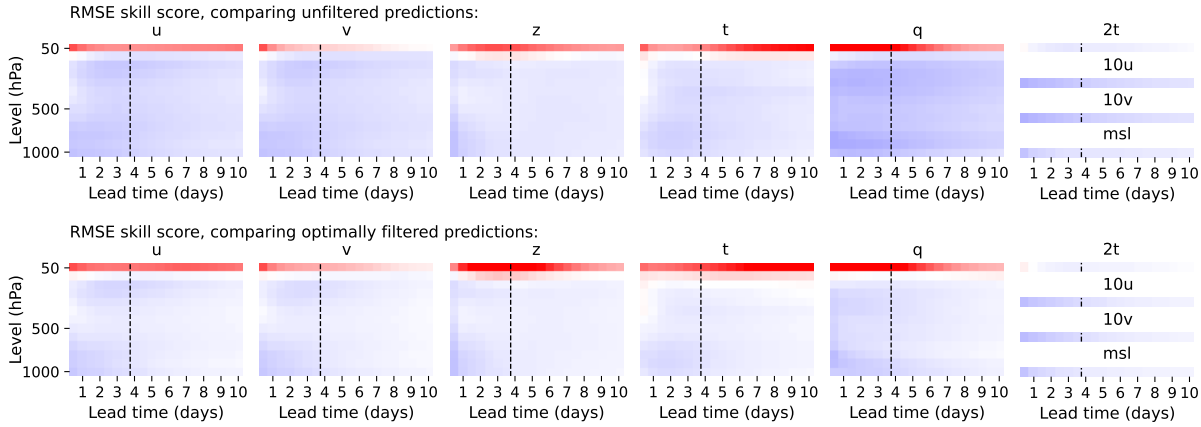

Fig. S32: **Effect of optimized blurring on GraphCast and HRES RMSE scorecards**. We show scorecards (as in Figure 2) comparing unfiltered predictions, and scorecards comparing optimally filtered predictions. In these scorecards each cell's color represents the RMSE skill score, where blue represents negative values (GraphCast has better skill) and red represents positive values (HRES has better skill).

pendent of the longitudinal wavenumber m. The filter weights were fitted using least-squares to minimize mean squared error, for each lead time separately, as computed in the spherical harmonic domain:

$$\mathcal{L}^{j,\tau}_{\text{filters}} = \frac{1}{|D_{\text{eval}}|} \sum_{d_0 \in D_{\text{eval}}} \frac{1}{4\pi} \sum_{l=0}^{l_{\max}} \sum_{m=-l}^{l} \left(b^{\tau}_{j,l} \hat{f}^{d_0+\tau}_{j,l,m} - f^{d_0+\tau}_{j,l,m} \right)^2. \tag{36}$$

We used data from 2017 to fit these weights, independently for GraphCast and HRES. This period does not overlap with the 2018 test set. When evaluating the filtered predictions, we computed MSE in the spherical harmonic domain, as detailed in Equation (22).

By fitting different filters for each lead time, the degree of blurring was free to increase with increasing uncertainty at longer lead times.

While this method is fairly general, it also has limitations. Because the filters are homogeneous, they are unable to take into account location-specific features, such as orography or land-sea boundaries, and so they must choose between over-blurring predictable high-resolution details in these locations, or under-blurring unpredictable high-resolution details more generally. This makes them less effective for some surface variables like 2T, which contain many

such predictable details. Future work may consider more complex post-processing schemes.

An alternative way to approximate a conditional expectation (and so improve RMSE) for our ECMWF forecast baseline would be to evaluate the ensemble mean of the ENS ensemble forecast system, instead of the deterministic HRES forecast. However the ENS ensemble is run at lower resolution than HRES, and because of this, it is unclear to us whether its ensemble mean will improve on the RMSE of a post-processed version of HRES. We leave an exploration of this for future work.

7.4.3 Transfer functions of the optimized filters

Since our optimized filters are linear, homogeneous and isotropic, they are uniquely identified by transfer functions (*74*) which specify their power gain (ratio of output power to input power) at each wavelength. We have plotted transfer functions for the optimized filters in Figure S33. Power gain is conventionally plotted on a logarithmic decibel scale; in our case the power gain in decibels is a simple transformation $20 \log_{10}(b^{\tau}_{j,l})$ of the filter weights $b^{\tau}_{j,l}$ from Equation (36).

The more negative the power gain over a range of wavelengths, the more power is attenuated, or the more signal is removed or blurred out, by the filter at these wavelengths. For both HRES and GraphCast, we see that the optimized filters generally attenuate power, or blur, over some short-to-mid wavelengths. As lead times increase from 12 hours to 10 days, the amount of blurring increases, and the blurring also starts to affect longer and longer wavelengths.

In optimizing for MSE, we seek to approximate a conditional expectation which averages over predictive uncertainty. Over longer lead times this predictive uncertainty increases, as does the spatial scale of uncertainty about the location of weather phenomena. We believe that this largely explains these changes in the optimized filter responses as a function of lead time.

We can see that the optimized filters tend to blur HRES more than GraphCast, because GraphCast's predictions already blur to some extent (see Section 7.5.3), whereas HRES' do

not.

The optimized filters are also able to compensate, to some extent, for spectral biases in the predictions of GraphCast and HRES. For example, for many variables in our regridded ERA5 dataset, the spectrum cuts off abruptly for wavelengths below 62km that are unresolved at ERA5's native $0.28125°$ resolution. GraphCast has not learned to replicate this cutoff exactly, but the optimal filters are able to implement it.

We also note that there are noticeable peaks in the GraphCast filter response around 100km wavelength for $\mathrm{Z500}$, which are not present for HRES. We believe these are filtering out small, spurious artifacts which are introduced by GraphCast around these wavelengths as a side-effect of the grid-to-mesh and mesh-to-grid transformations performed inside the model.

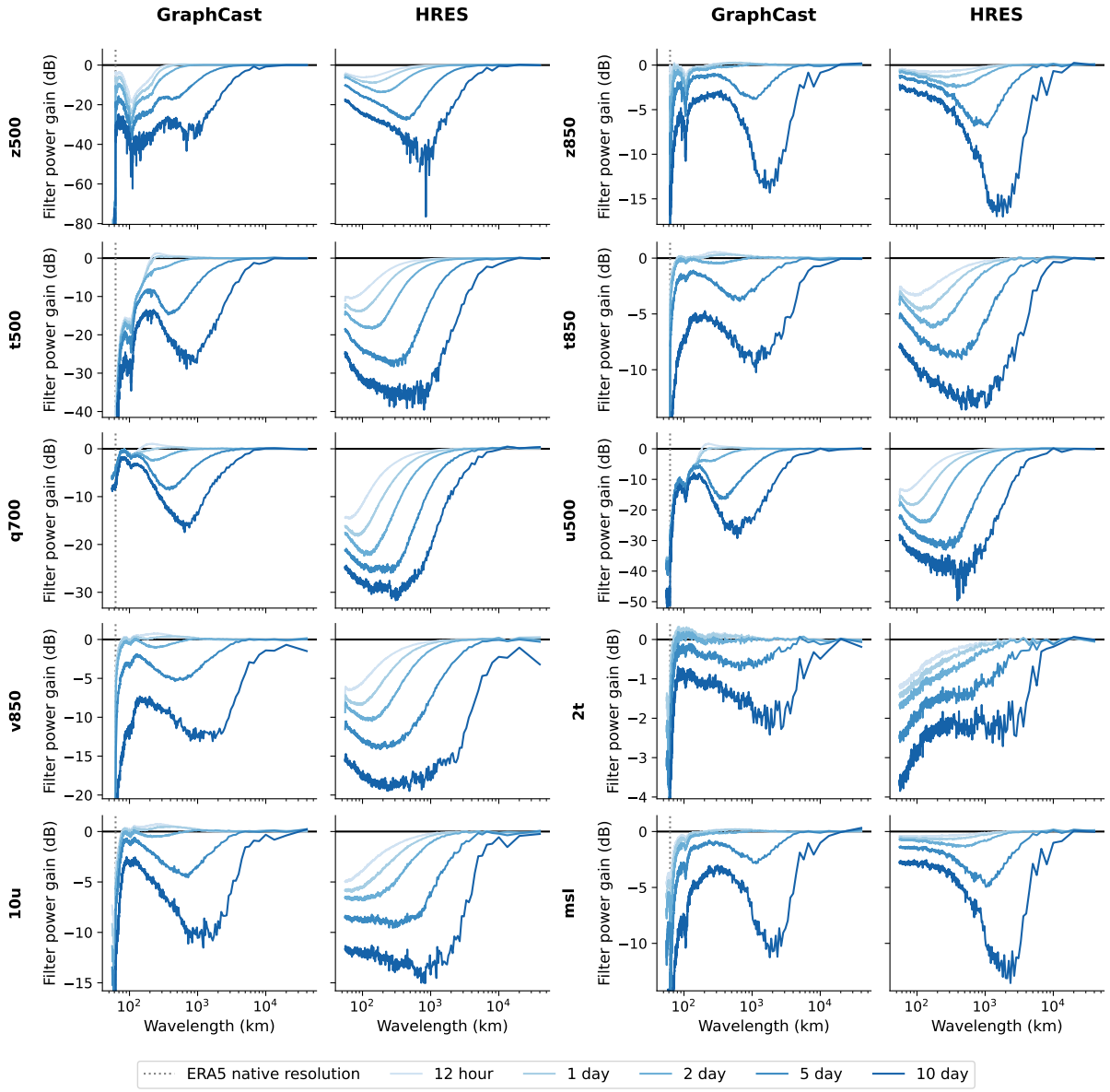

Fig. S33: **Transfer functions of optimized filters for GraphCast and HRES**. The y-axis shows the ratio of output power to input power for the filter, on the logarithmic decibel scale; lower values correspond to more attenuation of power or blurring at a given wavelength. This is plotted against wavelength on the x-axis. Blue lines correspond to filters fit for different lead times, and the horizontal black line at zero indicates an identity filter response. Vertical dotted lines on the GraphCast plots show the shortest wavelength (62km) resolved at ERA5's native resolution (TL639).

7.4.4 Relationship between autoregressive training horizon and blurring

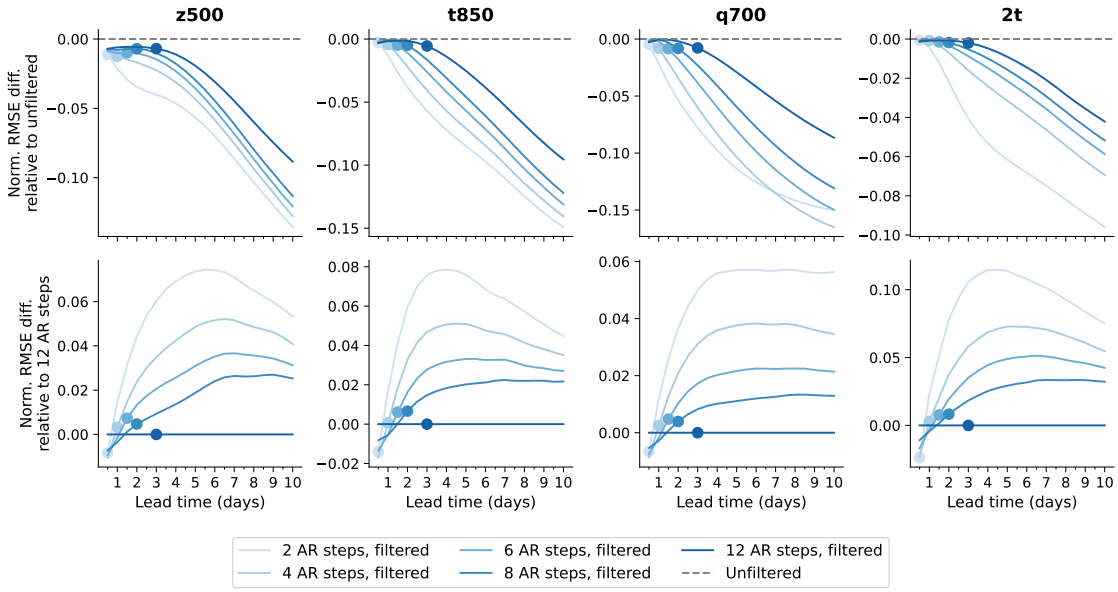

Fig. S34: **Results of optimized blurring, for GraphCast trained to different autoregressive training horizons**. In the first row we plot the RMSE of filtered predictions relative to corresponding unfiltered predictions. In the second row we plot the RMSE of filtered predictions relative to the filtered predictions trained to 12 autoregressive steps. Circles show the lead time equivalent to the autoregressive training horizon which each model was trained up to.

In Figure S34 we use the results of optimized blurring to investigate the connection between autoregressive training and the blurring of GraphCast's predictions at longer lead times.

In the first row of Figure S34, we see that models trained with longer autoregressive training horizons benefit less from optimized blurring, and that the benefits of optimized blurring generally start to accrue only *after* the lead time corresponding to the horizon they were trained up to. This suggests that autoregressive training is effective in teaching the model to blur up to the training horizon, but beyond this further blurring is required to minimize RMSE.

It would be convenient if we could replace longer-horizon training with a simple post-processing strategy like optimized blurring, but this does not appear to be the case: in the second row of Figure S34 we see that longer-horizon autoregressive training still results in

lower RMSEs, even after optimized blurring has been applied.

If one desires predictions which are in some sense minimally blurry, one could use a model trained to a small number of autoregressive steps. This would of course result in higher RMSEs at longer lead times, and our results here suggest that these higher RMSEs would not *only* be due to the lack of blurring; one would be compromising on other aspects of skill at longer lead times too. In some applications this may still be a worthwhile trade-off, however.

7.5 Spectral analysis

7.5.1 Spectral decomposition of mean squared error

In Figure S35 we compare the skill of GraphCast with HRES over a range of spatial scales, by plotting the contribution of each wavelength towards the mean squared error for both models. In Figure S36 we have also repeated this analysis after the optimized blurring described in Section 7.4.

The MSE, via its spectral formulation (Equation (22)) can be decomposed as a sum of mean error powers at different total wavenumbers:

$$\text{MSE}_{sh}(j,\tau) = \sum_{l=0}^{l_{\text{max}}} S^{j,\tau}(l) \tag{37}$$

$$S^{j,\tau}(l) = \frac{1}{|D_{\text{eval}}|} \sum_{d_0 \in D_{\text{eval}}} \frac{1}{4\pi} \sum_{m=-l}^{l} \left(\hat{f}_{j,l,m}^{d_0+\tau} - f_{j,l,m}^{d_0+\tau} \right)^2, \tag{38}$$

where $l_{\text{max}} = 719$ as in Equation (22). Each total wavenumber l corresponds approximately to a wavelength C_e/l, where C_e is the earth's circumference. We plot $S^{j,\tau}(l)$ as a function of wavelength C_e/l on a log-log scale. As a note of caution, because of the log-log scale, area under the curve does not correspond to MSE in these plots.

At lead times of 2 days or more, for the majority of variables GraphCast improves on the skill of HRES uniformly over all wavelengths. (2m temperature is a notable exception).

At shorter lead times of 12 hours to 1 day, for a number of variables (including Z500, T500, T850 and U500) HRES has greater skill than GraphCast at scales in the approximate range of 200-2000km, with GraphCast generally having greater skill outside this range.

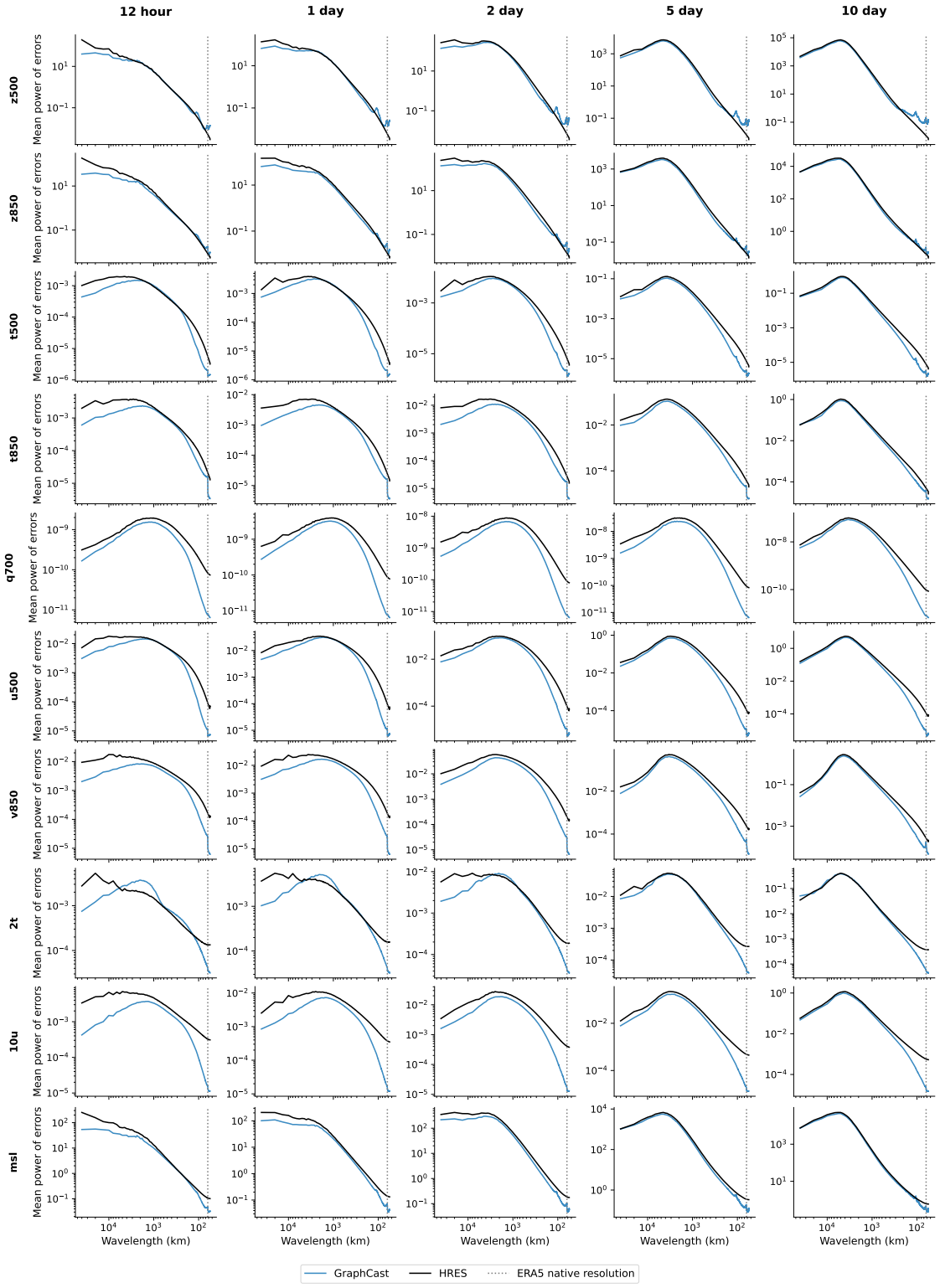

Fig. S35: **Spectral decomposition of mean squared error for GraphCast and HRES.** We plot the mean power spectrum of errors as a function of wavelength on a log-log scale. Dotted vertical lines indicate the native resolution of GraphCast's ERA5 training data.

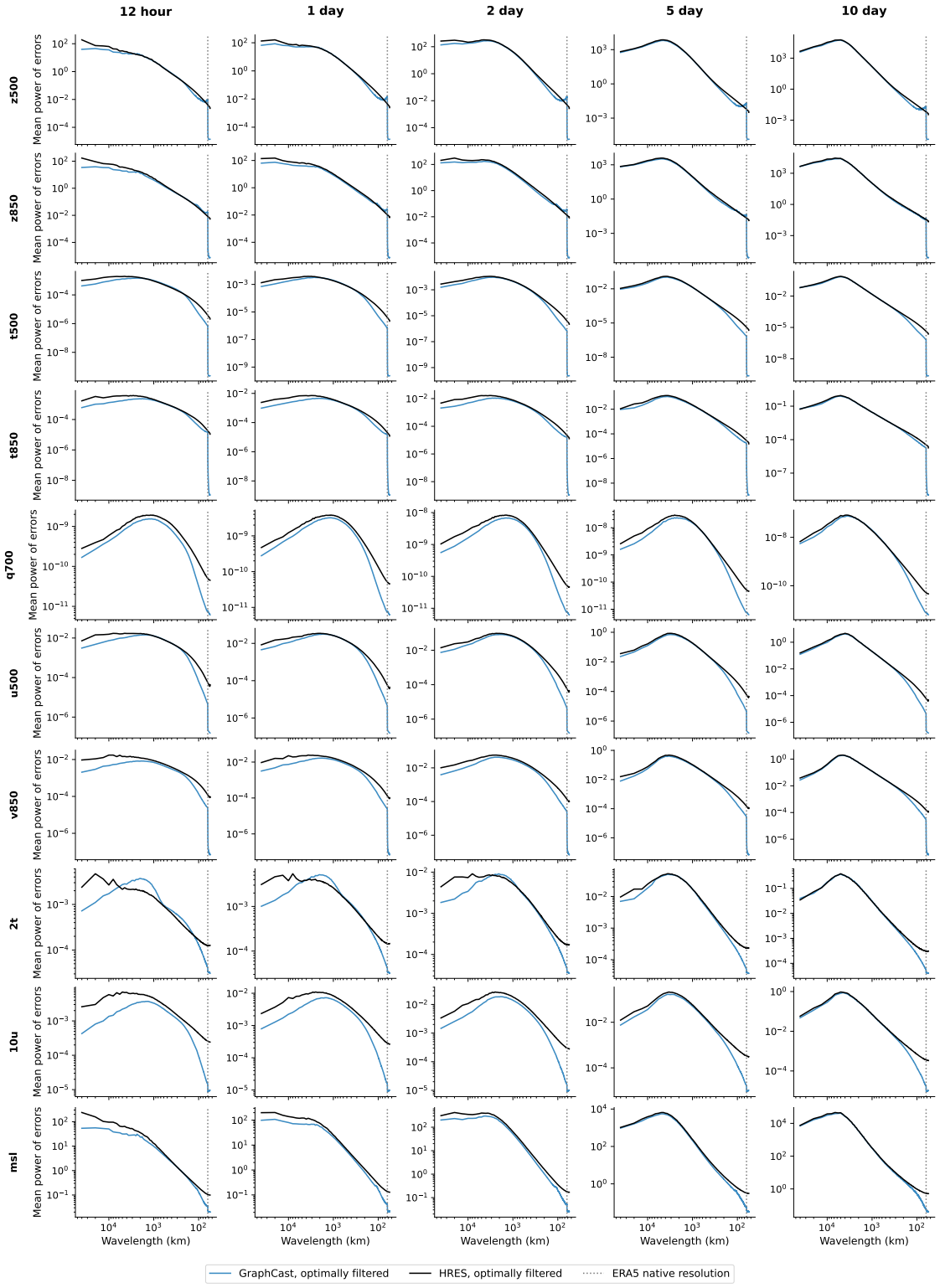

Fig. S36: **Spectral decomposition of mean squared error for GraphCast and HRES after optimized blurring.** We plot the mean power spectrum of errors as a function of wavelength on a log-log scale. Dotted vertical lines indicate the native resolution of GraphCast's ERA5 training data.

7.5.2 RMSE as a function of horizontal resolution

In Figure S37, we compare the skill of GraphCast with HRES when evaluated at a range of spatial resolutions. Specifically, at each total wavenumber l_{trunc}, we plot RMSEs between predictions and targets which are both truncated at that total wavenumber. This is approximately equivalent to a wavelength C_e/l_{trunc} where C_e is the earth's circumference.

The RMSEs between truncated predictions and targets can be obtained via cumulative sums of the mean error powers $S^{j,\tau}(l)$ defined in Equation (38), according to

$$\text{RMSE}_{\text{trunc}}(j, \tau, l_{\text{trunc}}) = \sqrt{\sum_{l=0}^{l_{\text{trunc}}} S^{j,\tau}(l)}. \tag{39}$$

Figure S37 shows that in most cases GraphCast has lower RMSE than HRES at all resolutions typically used for forecast verification. This applies before and after optimized blurring (see Section 7.4). Exceptions include 2 meter temperature at a number of lead times and resolutions, T500 at 12 hour lead times, and U500 at 12 hour lead times, where GraphCast does better at full resolution but HRES does better at resolutions corresponding to shortest wavelengths of around 100 to 500 km.

In particular we note that the native resolution of ERA5 is $0.28125°$ corresponding to a shortest wavelength of 62km, indicated by a vertical line in the plots. HRES-fc0 targets contain some signal at wavelengths shorter than 62km, but the ERA5 targets used to evaluate GraphCast do not, natively at least (see Section 7.5.3). In Figure S37 we can see that evaluating at $0.28125°$ resolution instead of 0.25° does not significantly affect the comparison of skill between GraphCast and HRES.

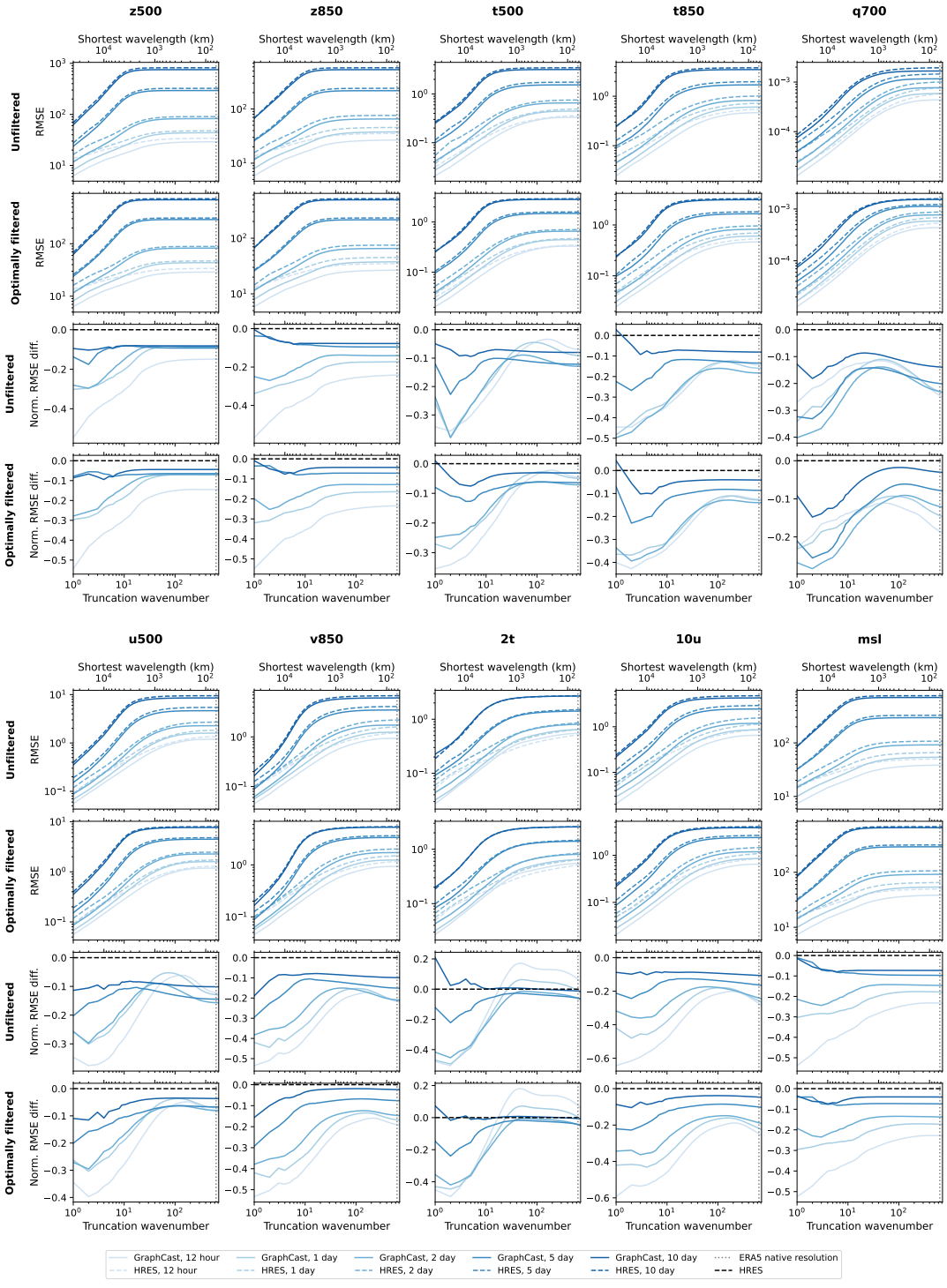

Fig. S37: **RMSE as a function of horizontal resolution**. The x-axes show the total wavenumber and wavelength at which both predictions and targets were truncated. The y-axes show RMSEs (rows 1,2,5,6), and RMSE skill scores relative to HRES (rows 3,4,7,8). We give results before optimized blurring (rows 1,3,5,7) and after (rows 2,4,6,8).

7.5.3 Spectra of predictions and targets

Figure S38 compares the power spectra of GraphCast's predictions, the ERA5 targets they were trained against, and HRES-fc0. A few phenomena are notable:

Differences between HRES and ERA5 There are noticeable differences in the spectra of ERA5 and HRES-fc0, especially at short wavelengths. These differences may in part be caused by the methods used to regrid them from their respective native IFS resolutions of TL639 ($0.28125°$) and TCo1279 (approx. $0.1°$, (*40*)) to a 0.25° equiangular grid. However even before this regridding is done there are differences in IFS versions, settings, resolution and data assimilation methodology used for HRES and ERA5, and these differences may also affect the spectra. Since we evaluate GraphCast against ERA5 and HRES against HRES-fc0, this domain gap remains an important caveat to attach to our conclusions.

Blurring in GraphCast We see reduced power at short-to-mid wavelengths in GraphCast's predictions which reduces further with lead time. We believe this corresponds to blurring which GraphCast has learned to perform in optimizing for MSE. We discussed this further in Sections 7.4 and 7.4.4.

Peaks for GraphCast around 100km wavelengths These peaks are particularly visible for z500; they appear to increase with lead time. We believe they correspond to small, spurious artifacts introduced by the internal grid-to-mesh and mesh-to-grid transformations performed by GraphCast at each autoregressive step. In future work we hope to eliminate or reduce the effect of these artifacts, which were also observed by (*13*).

Finally we note that, while these differences in power at short wavelengths are very noticeable in log scale and relative plots, these short wavelengths contribute little to the total power

of the signal.

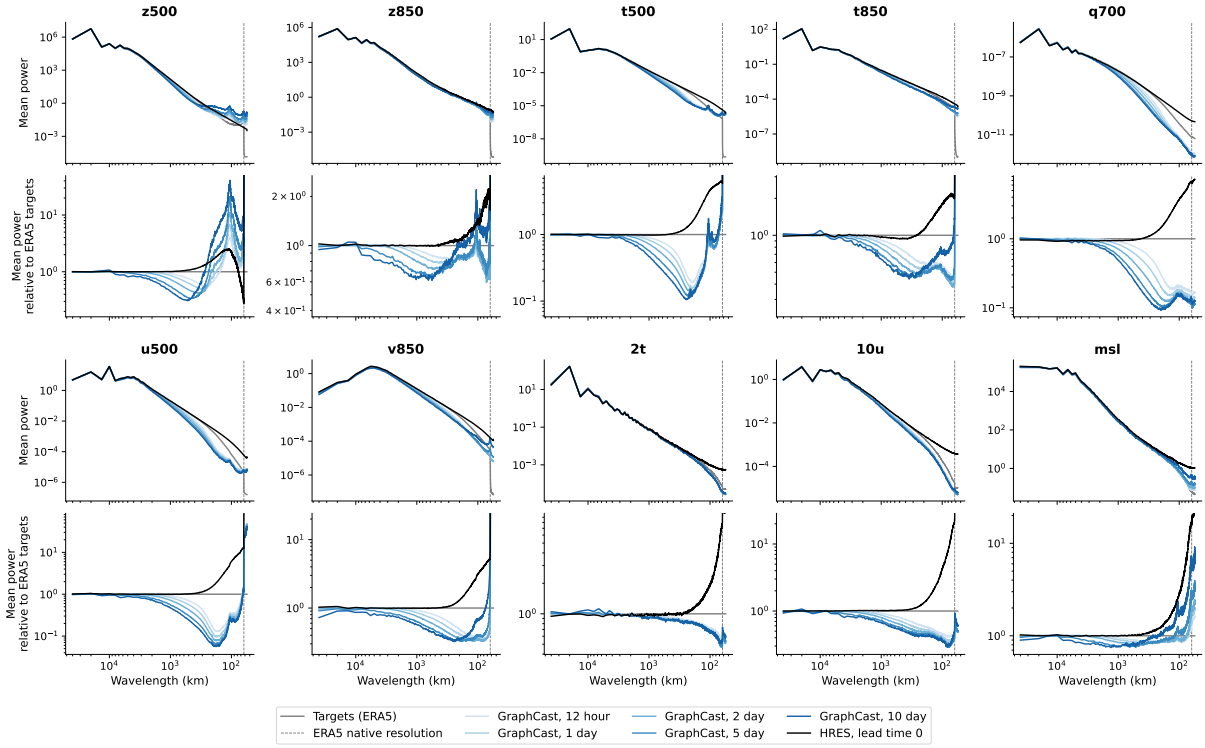

Fig. S38: **Power spectra of predictions and targets for GraphCast**. For each variable, the first row plots power at each total wavenumber on a log-log scale; the second row plots power relative to the ERA5 targets used for GraphCast. We also show the spectrum of HRES in black.

8 Additional severe event forecasting results

In this section, we provide additional details about our severe event forecasting analysis. We note that GraphCast is not specifically trained for those downstream tasks, which demonstrates that, beyond improved skills, GraphCast provides useful forecast for tasks with real-world impact such as tracking cyclones (Section 8.1), characterizing atmospheric rivers (Section 8.2), and classifying extreme temperature (Section 8.3). Each task can also be seen as evaluating the value of GraphCast on a different axis: spatial and temporal structure of high-resolution prediction (cyclone tracking task), ability to non-linearly combine GraphCast predictions to derive quantities of interest (atmospheric rivers task), and ability to characterize extreme and rare events (extreme temperatures).

8.1 Tropical cyclone track forecasting

In this section, we detail the evaluation protocols we used for cyclone tracking (Supplements Section 8.1.1) and analyzing statistical significance (Supplements Section 8.1.2), provide additional results (Supplements Section 8.1.3), and describe our tracker and its differences with the one from ECMWF (Supplements Section 8.1.4).

8.1.1 Evaluation protocol

The standard way of comparing two tropical cyclone prediction systems is to restrict the comparison to events where both models predict the existence of a cyclone. As detailed in Supplements Section 5.2.2, GraphCast is initialized from 06z and 18z, rather than 00z and 12z, to avoid giving it a lookahead advantage over HRES. However, the HRES cyclone tracks in the TIGGE archive (*27*) are only initialized at 00z and 12z. This discrepancy prevents us from selecting events where the initialization and lead time map to the same validity time for both methods, as there is always a 6h mismatch. Instead, to compare HRES and GraphCast on a set of similar

events, we proceed as follows. We consider all the dates and times for which our ground truth dataset IBTrACS (*29, 30*) identified the presence of a cyclone. For each cyclone, if its time is 06z or 18z, we make a prediction with GraphCast starting from that date, apply our tracker and keep all the lead times for which our tracker detects a cyclone. Then, for each initialization time/lead time pairs kept for GraphCast, we consider the two valid times at +/-6h around the initialization time of GraphCast, and use those as initialization time to pick the corresponding HRES track from the TIGGE archive. If, for the same lead time as GraphCast, HRES detects a cyclone, we include both GraphCast and HRES initialization time/lead time pairs into the final set of events we use to compare them. For both methods, we only consider predictions up to 120 hours.

Because we compute error with respect to the same ground truth (i.e., IBTrACS), the evaluation is not subject to the same restrictions described in Supplements Section 5.2.2, i.e., the targets for both models incorporate the same amount of lookahead. This is in contrast with most our evaluations in this paper, where the targets for HRES (i.e., HRES-fc0) incorporates +3h lookahead, and the ones for GraphCast (from ERA5) incorporate +3h or +9h, leading us to only report results for the lead times with a matching lookahead (multiples of 12h). Here, since the IBTrACS targets are the same for both models, we can report performance as a function of lead time by increments of 6h.

For a given forecast, the error between the predicted center of the cyclone and the true center is computed using the geodesic distance.

8.1.2 Statistical methodology

Computing statistical confidence in cyclone tracking requires particular attention in two aspects:

1. There are two ways to define the number of samples. The first one is the number of tropical cyclone events, which can be assumed to be mostly independent events. The

second one is the number of per-lead time data points used, which is larger, but accounts for correlated points (for each tropical cyclone event multiple predictions are made at 6h interval). We chose to use the first definition which provides more conservative estimates of statistical significance. Both numbers are shown for lead times 1 to 5 days on the x-axis of Supplements Figure S39.

2. The per-example tracking errors of HRES and GraphCast are correlated. Therefore statistical variance in their difference is much smaller than their joint variance. Thus, we report the confidence that GraphCast is better than HRES (see Supplements Figure S39b) in addition to the per-model confidence (see Supplements Figure S39a).

Given the two considerations above, we do bootstrapping with 95% confidence intervals at the level of cyclones. For a given lead time, we consider all the corresponding initialization time/lead time pairs and keep a list of which cyclone they come from (without duplication). For the bootstrap estimate, we draw samples from this cyclone list (with replacement) and apply the median (or the mean) to the corresponding initialization time/lead time pairs. Note that this gives us much more conservative confidence bounds than doing bootstrapping at the level of initialization time/lead time pairs, as it is equivalent to assuming all bootstrap samples coming from the sample cyclone (usually in the order of tens) are perfectly correlated.

For instance, assume for a given lead time we have errors of (50, 100, 150) for cyclone A, (300, 200) for cyclone B and (100, 100) for cyclone C, with A having more samples. A bootstrapping sample at the level of cyclones first samples uniformly at random 3 cyclones with replacement (for instance A,A,B) and then computes the mean on top of the corresponding samples with multiplicity: mean(50,100,150,50,100,150,200,300)=137.5.

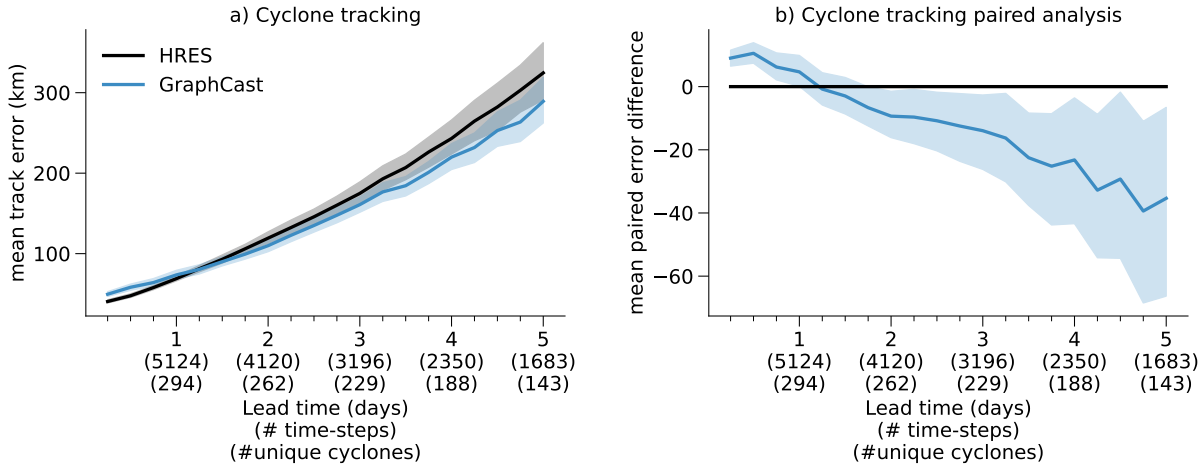

Fig. S39: **Mean performance on cyclone tracking *(lower is better)*** a) Cyclone tracking performances for GraphCast and HRES. The x-axis represents lead times (in days). The y-axis represents mean track error (in km). The error bars represent the bootstrapped error of the mean. b) Paired analysis of cyclone tracking. The x-axis represents lead times (in days). The y-axis represents mean per-track error difference between HRES and GraphCast. The error bars represent the bootstrapped error of the mean.

8.1.3 Results

In Figure 3a-b, we chose to show the median error rather than the mean. This decision was made before computing the results on the test set, based on the performance on the validation set. On the years 2016–2017, using the version of GraphCast trained on 1979–2015, we observed that, using early versions of our tracker, the mean track error was dominated by very few outliers and was not representative of the overall population. Furthermore, a sizable fraction of these outliers were due to errors in the tracking algorithm rather than the predictions themselves, suggesting that the tracker was suboptimal for use with GraphCast. Because our goal is to assess the value of GraphCast forecast, rather than a specific tracker, we show median values, which are also affected by tracking errors, but to a lesser extent. In figure Figure S40 we show how that the distribution of both HRES and GraphCast track errors for the test years 2018–2021 are non-gaussian with many outliers. This suggests the median is a better summary statistic than the mean.

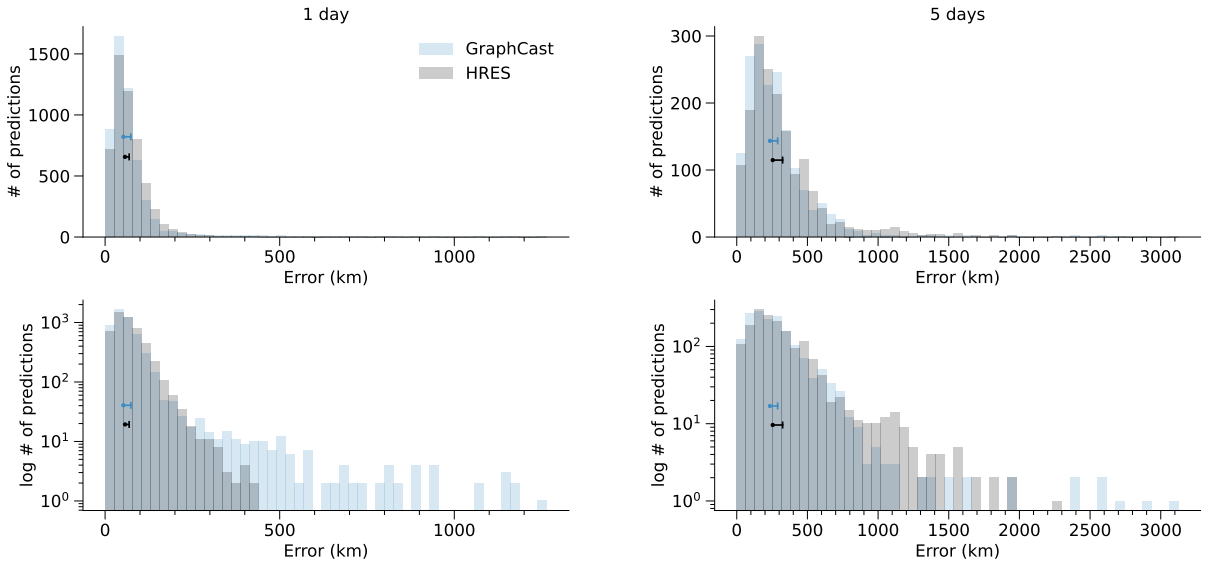

Fig. S40: **Histograms of cyclone track errors** with linear and logarithmic y-axis. The horizontal lines connect the median error (circle) to the mean error (vertical tick) for each model. We observe that the distribution of errors of HRES, and particularly GraphCast, have heavier tails than a Gaussian, breaking the implicit Gaussian assumption of the RMSE estimator.

Supplements Figure S39 complements Figure 3a-b by showing the mean track error and the corresponding paired analysis. We note that using the final version of our tracker (Supplements Section 8.1.4), GraphCast mean results are similar to the median one, with GraphCast significantly outperforming HRES for lead time between 2 and 5 days.

Because of well-known blurring effects, which tend to smooth the extrema used by a tracker to detect the presence of a cyclone, ML methods can drop existing cyclones more often than NWPs. Dropping a cyclone is very correlated with having a large positional error. Therefore, removing from the evaluation such predictions, where a ML model would have performed particularly poorly, could give it an unfair advantage.

To avoid this issue, we verify that our hyper-parameter-searched tracker (see Supplements Section 8.1.4) misses a similar number of cyclones as HRES. Supplements Figure S41 shows that on the test set (2018–2021), GraphCast and HRES drop a similar number of cyclones, ensuring

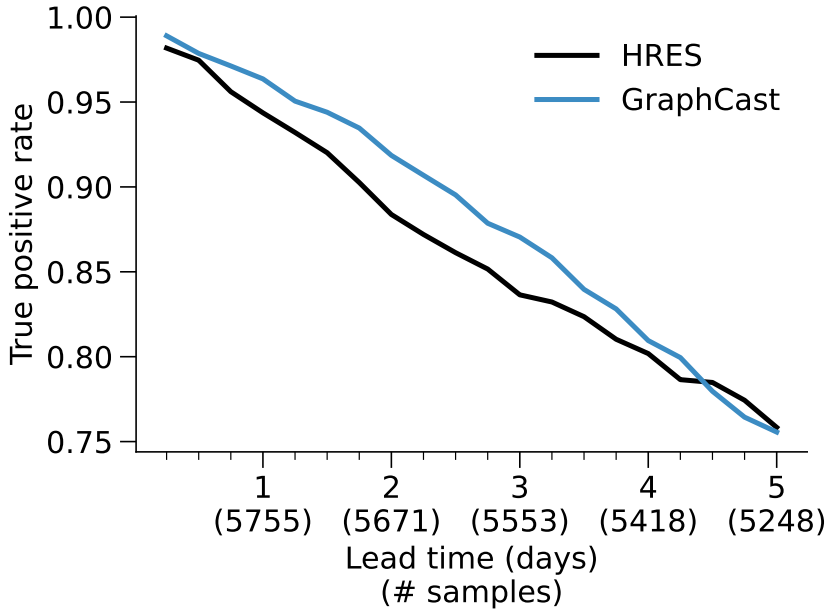

Fig. S41: **True positive rate detection of cyclones *(higher is better)*** GraphCast and HRES detect a comparable number of cyclones, decreasing as a function of lead time.

our comparisons are as fair as possible.

Supplements Figures S42 and S43 show the median error and paired analysis as a function of lead time, broken down by cyclone category, where category is defined on the Saffir-Simpson Hurricane Wind Scale (*75*), with category 5 representing the strongest and most damaging storms (note, we use category 0 to represent tropical storms). We found that GraphCast has equivalent or better performance than HRES across all categories, although the sample sizes are smaller. For category 5, the most intense events, GraphCast is significantly better that HRES for lead times beyond two days, as demonstrated by the per-track paired analysis. We also obtain similar results when measuring mean performance instead of median.

Another important part of cyclone forecasting is to characterize their intensity. However, even point-wise data from ERA5 at 1h-0.25° resolution poorly correlates to the category of a cyclone, with events corresponding to both low and high categories resulting in speeds of about 60-110km/h, well below the 1min-wind speed thresholds that define the Saffir-Simpson scale

(119km/h for category 1, 252km/h for category 5). We thus leave predicting the category from GraphCast forecasts for future work.

8.1.4 Tracker details

The tracker we used for GraphCast is based on our reimplementation of ECMWF's tracker (*26*). Because it is designed for 0.1° HRES, we found it helpful to add several modifications to reduce the amount of mistracked cyclones when applied to GraphCast predictions. However, tracking errors still occur, which is expected from tracking cyclone from 0.25° predictions instead of 0.1°. We note that we do not use our tracker for the HRES baseline, as its tracks are directly recovered from the TIGGE archives (*27*).

We first give a high-level summary of the default tracker from ECMWF, before explaining the modifications we made and our decision process.

ECMWF tracker Given a model's predictions of the variables 10U, 10V, MSL as well as U, V and Z at pressure levels 200, 500, 700, 850 and 1000 hPa over multiple time steps, the ECMWF tracker (*35*) sequentially processes each time step to iteratively predict the location of a cyclone over an entire trajectory. Each 6h prediction of the tracker has two main steps. In the first step, based on the current location of the cyclone, the tracker computes an estimate of the next location, 6h ahead. The second step consists in looking in the vicinity of that new estimate for locations that satisfy several conditions that are characteristic of cyclone centers.

To compute the estimate of the next cyclone location, the tracker moves the current estimate using a displacement computed as the average of two vectors: 1) the displacement between the last two track locations (i.e., linear extrapolation) and 2) an estimate of the wind steering, averaging the wind speed U and V at the previous track position at pressure levels 200, 500, 700 and 850 hPa.

Once the estimate of the next cyclone location is computed, the tracker looks at all local

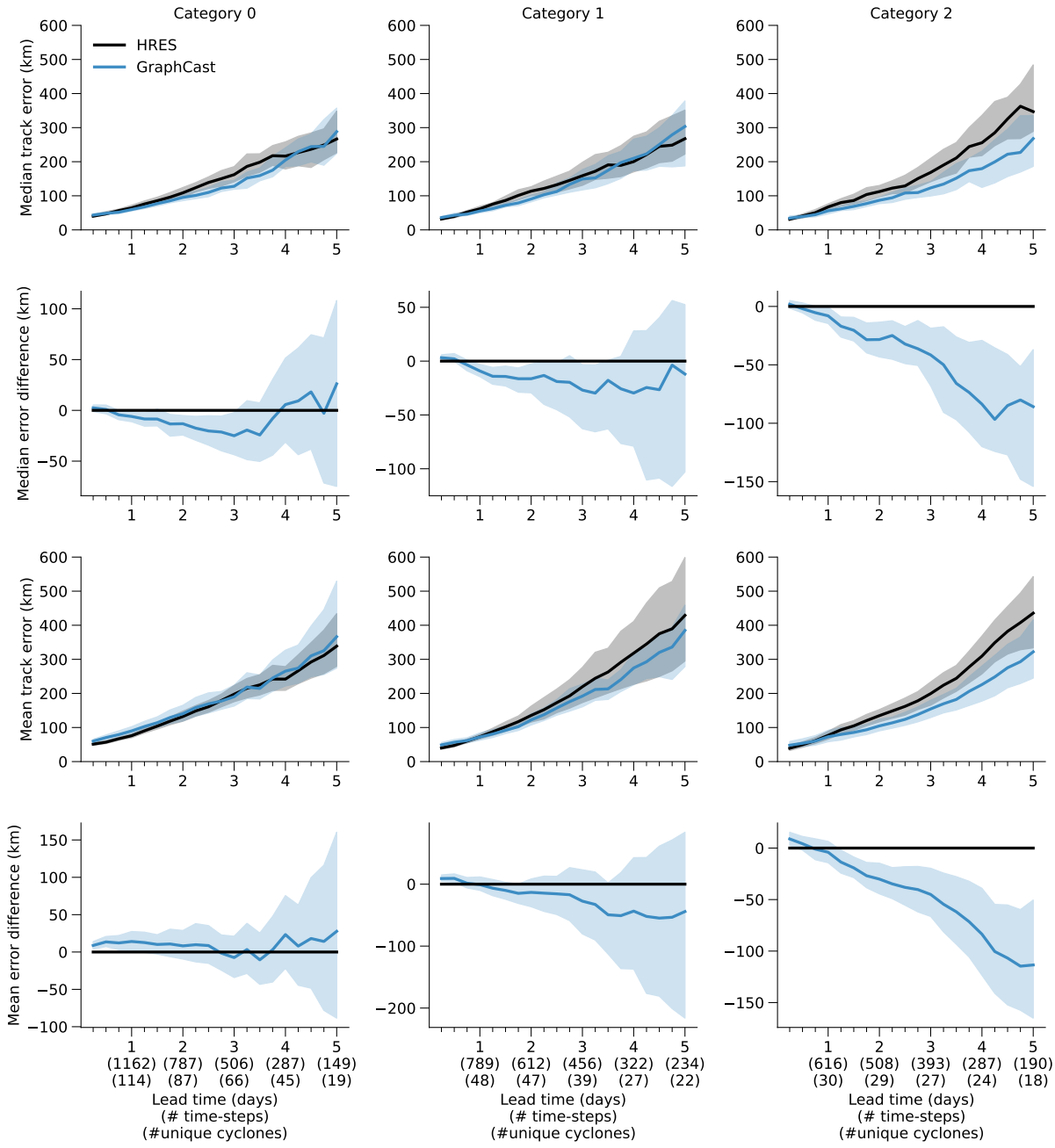

Fig. S42: **Per-cyclone-category median and mean performance (category 0 to 2)** Each column corresponds to a cyclone category from 0 to 2 on the Saffir-Simpson Hurricane Wind Scale.

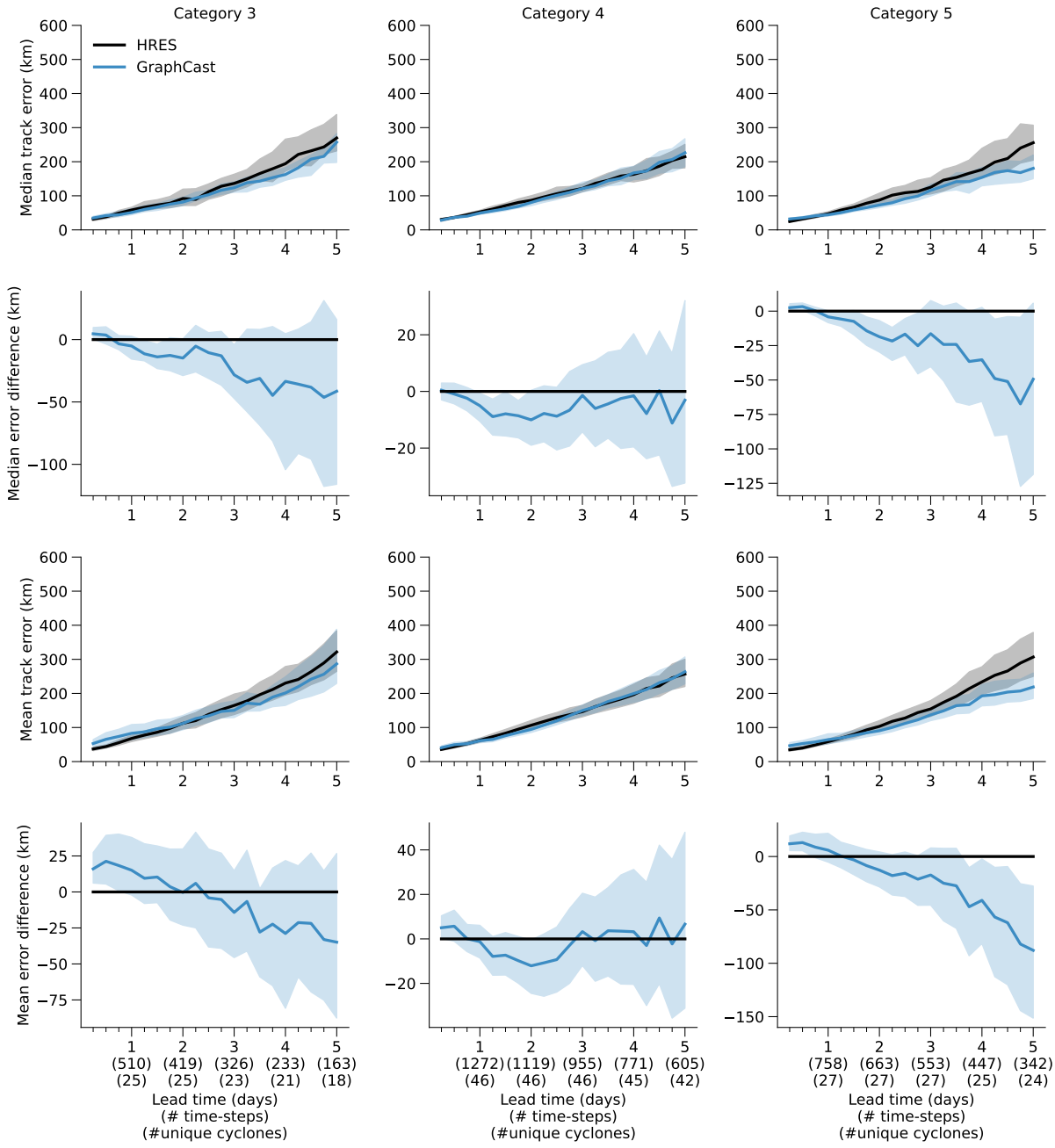

Fig. S43: **Per-cyclone-category median and mean performance (category 3 to 5)** Each column corresponds to a cyclone category from 3 to 5 on the Saffir-Simpson Hurricane Wind Scale.

minima of mean sea-level pressure (MSL) within 445 km of this estimate. It then searches for the candidate minima closest to the current estimate that satisfies the following three conditions:

1. Vorticity check: the maximum vorticity at 850 hPa within 278 km of the local minima is larger than $5 \cdot 10^{-5}$ s^{-1} for the Northern Hemisphere, or is smaller than $-5 \cdot 10^{-5} s^{-1}$ for the Southern Hemisphere. Vorticity can be derived from horizontal wind (U and V).

2. Wind speed check: if the candidate is on land, the maximum 10m wind speed within 278 km is larger than 8 m/s.

3. Thickness check: if the cyclone is extratropical, there is a maximum of thickness between 850 hPa and 200 hPa within a radius of 278 km, where the thickness is defined as Z850-Z200.

If no minima satisfies all those conditions, the tracker considers that there is no cyclone. ECMWF's tracker allows cyclones to briefly disappear under some corner-case conditions before reappearing. In our experiment with GraphCast, however, when a cyclone disappear, we stop the tracking.

Our modified tracker We analysed the mistracks on cyclones from our validation set years (2016–2017), using a version of GraphCast trained on 1979–2015, and modified the default re-implementation of the ECMWF tracker as described below. When we conducted a hyper-parameter search over the value of a parameter, we marked in bold the values we selected.

1. The current step vicinity radius determines how far away from the estimate a new center candidate can be. We found this parameter to be critical and searched a better value among the following options: $445 \times f$ for f in 0.25, 0.375, **0.5**, 0.625, 0.75, 1.0 (original value).

2. The next step vicinity radius determines how strict multiple checks are. We also found this parameter to be critical and searched a better value among the following options: $278 \times f$ for f in 0.25, 0.375, 0.5, 0.625, **0.75**, 1.0 (original value).

3. The next-step estimate of ECMWF uses a 50-50 weighting between linear extrapolation and wind steering vectors. In our case where wind is predicted at 0.25° resolution, we found wind steering to sometimes hinder estimates. This is not surprising because the wind is not a spatially smooth field, and the tracker is likely tailored to leverage 0.1° resolution predictions. Thus, we hyper-parameter searched the weighting among the following options: 0.0, 0.1, 0.33, **0.5** (original value).

4. We noticed multiple misstracks happened when the track sharply reversed course, going against its previous direction. Thus, we only consider candidates that creates an angle between the previous and new direction below d degrees, where d was searched among these values: **90**, 135, 150, 165, 175, 180 (i.e. no filter, original value).

5. We noticed multiple misstracks made large jumps, due to a combination of noisy wind steering and features being hard to discern for weak cyclones. Thus, we explored clipping the estimate from moving beyond x kilometers (by resizing the delta with the last center), searching over the following values for x: $445 \times f$ for f in 0.25, 0.5, 1.0, 2.0, 4.0, ∞ (i.e. no clipping, original value).

During the hyper-parameter search, we also verified on validation data that the tracker applied to GraphCast dropped a similar number of cyclones as HRES.

Note, we also ran an identical hyper-parameter search for our tracker applied to HRES predictions at 0.25° resolution. We found that, on the validation years and at equal true positive rate, the performance gap between HRES and GraphCast was larger when using our tracker

(optimized for HRES) than when retrieving the HRES tracks from the TIGGE archive. This supports using the HRES tracks from TIGGE in our evaluations.

8.2 Atmospheric rivers

The vertically integrated water vapor transport (IVT) is commonly used to characterize the intensity of atmospheric rivers (*32, 33*). Although GraphCast does not directly predict IVT and is not specifically trained to predict atmospheric rivers, we can derive this quantity from the predicted atmospheric variables specific humidity, Q, and horizontal wind, (U, V), via the relation (*32*):

$$IVT = \frac{1}{g}\sqrt{\left(\int_{p_b}^{p_t} \mathrm{Q}(p)\mathrm{U}(p)dp\right)^2 + \left(\int_{p_b}^{p_t} \mathrm{Q}(p)\mathrm{V}(p)dp\right)^2}, \tag{40}$$

where $g = 9.80665\ \mathrm{m/s^2}$ is the acceleration due to gravity at the surface of the Earth, $p_b = 1000\ \mathrm{hPa}$ is the bottom pressure, and $p_t = 300\ \mathrm{hPa}$ is the top pressure.

Evaluation of IVT using the above relation requires numerical integration and the result therefore depends on the vertical resolution of the prediction. GraphCast has a vertical resolution of 37 pressure levels which is higher than the resolution of the available HRES trajectories with only 25 pressure levels. For a consistent and fair comparison of both models, we therefore only use a common subset of pressure levels, which are also included in the WeatherBench benchmark, when evaluating IVT [13], namely $[300, 400, 500, 600, 700, 850, 925, 1000]\ \mathrm{hPa}$.

Consistently with the rest of our evaluation protocol, each model is evaluated against its own "analysis". For GraphCast, we compute the IVT based on its predictions and we compare it to the IVT computed analogously from ERA5. Similarly, we use HRES predictions to compute the IVT for HRES and and compare it to the IVT computed from HRES-fc0.

Similarly to previous work (*31*), Figure S44 reports RMSE skill and skill score averaged over coastal North America and the Eastern Pacific (from 180°W to 110°W longitude, and 10°N

[13] As suggested by ECMWF during personal conversation.

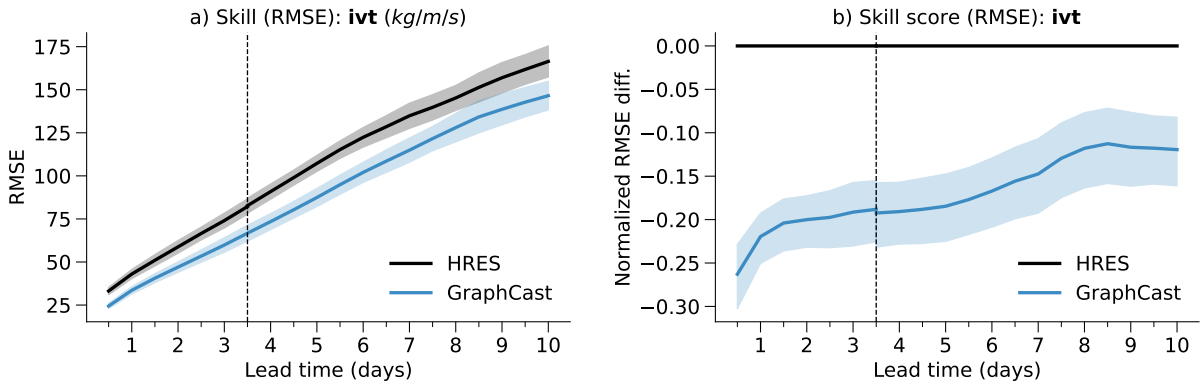

Fig. S44: **Skill and skill score for GraphCast and HRES on vertically integrated water vapor transport (ivt) *(lower is better)*.** (a) RMSE skill (y-axis) for GraphCast (blue line) and HRES (black line) on IVT as a function of lead time (x-axis), with 95% confidence interval error bars (see Section 5.4.3). (b) RMSE skill score (y-axis) for GraphCast and HRES with respect to HRES on IVT as a function of lead time (x-axis), with 95% confidence interval error bars (see Section 5.4.4). GraphCast improves the prediction of IVT compared to HRES, from 25% at short lead time, to 10% at longer horizon.

to 60°N latitude) during the cold season (Jan-April and Oct-Dec 2018), which corresponds to a region and a period with frequent atmospheric rivers.

8.3 Extreme heat and cold

We study extreme heat and cold forecasting as a binary classification problem (*3, 35*) by comparing whether a given forecasting model can correctly predict whether the value for a certain variable will be above (or below) a certain percentile of the distribution of a reference historical climatology (for example above 98% percentile for extreme heat, and below 2% percentile for extreme cold). Following previous work (*35*), the reference climatology is obtained separately for (1) each variable, (2) each month of the year, (3) each time of the day, (4) each latitude/longitude coordinate, and (5) each pressure level (if applicable). This makes the detection of extremes more contrasted by removing the effect of the diurnal and seasonal cycles in each spatial location. To keep the comparison as fair as possible between HRES and GraphCast, we compute this climatology from HRES-fc0 and ERA5 respectively, for years 2016-2021. We

experimented with other ways to compute climatology (2016-2017 as well as using ERA5 climatology 1993-2016 for both models), and found that results hold generally.

Because extreme prediction is by definition an imbalanced classification problem, we base our analysis on precision-recall plots which are well-suited for this case (*37*). Precision and recall are defined respectively as,

$$\mathrm{P} = \frac{tp}{tp + fp} \tag{41}$$

$$\mathrm{R} = \frac{tp}{tp + fn}, \tag{42}$$

where tp, fp, and fn, indicate "true positives", "false positives" and "false negatives", respectively. The precision-recall curve is obtained by varying a free parameter "gain" consisting of a scaling factor with respect to the median value of the climatology, i.e. scaled forecast $=$ gain $\times$ (forecast $-$ median climatology) $+$ median climatology. This has the effect of shifting the decision boundary and allows to study different trade offs between false negatives and false positives. Intuitively, a 0 gain will produce zero forecast positives (e.g. zero false positives), and an infinite gain will produce amplify every value above the median to be a positive (so potentially up to 50% false positive rate). The "gain" is varied smoothly from 0.8 to 4.5. Similar to the rest of the results in the paper we also use labels from HRES-fc0 and ERA5 when evaluating HRES and GraphCast, respectively.

We focus our analysis on variables that are relevant for extreme temperature conditions, specifically 2T (*3, 35*), and also T850, Z500 which are often used by ECMWF to characterize heatwaves (*36*). Following previous work (*3*), for extreme heat we average across June, July, and August over land in the northern hemisphere (latitude > 20°) and across December, January, and February over land in the southern hemisphere (latitude < -20°). For extreme cold, we swapped the months for the northern and southern hemispheres. See full results in Figure S45. We also provide a more fine-grained lead-time comparison, by summarizing the precision-recall

curves by selecting the point with the highest SEDI score (*35*) and showing this as function of lead time (Figure S46). The SEDI score is defined as,

$$\text{SEDI} = \frac{\log F - \log R - \log(1-F) + \log(1-R)}{\log F + \log R + \log(1-F) + \log(1-R)}, \tag{43}$$

where R is the recall, and F is the "false positive rate", defined as,

$$\text{F} = \frac{fp}{fp + tn}, \tag{44}$$

where fp, and tn, indicate "false positives", "true negatives", respectively.

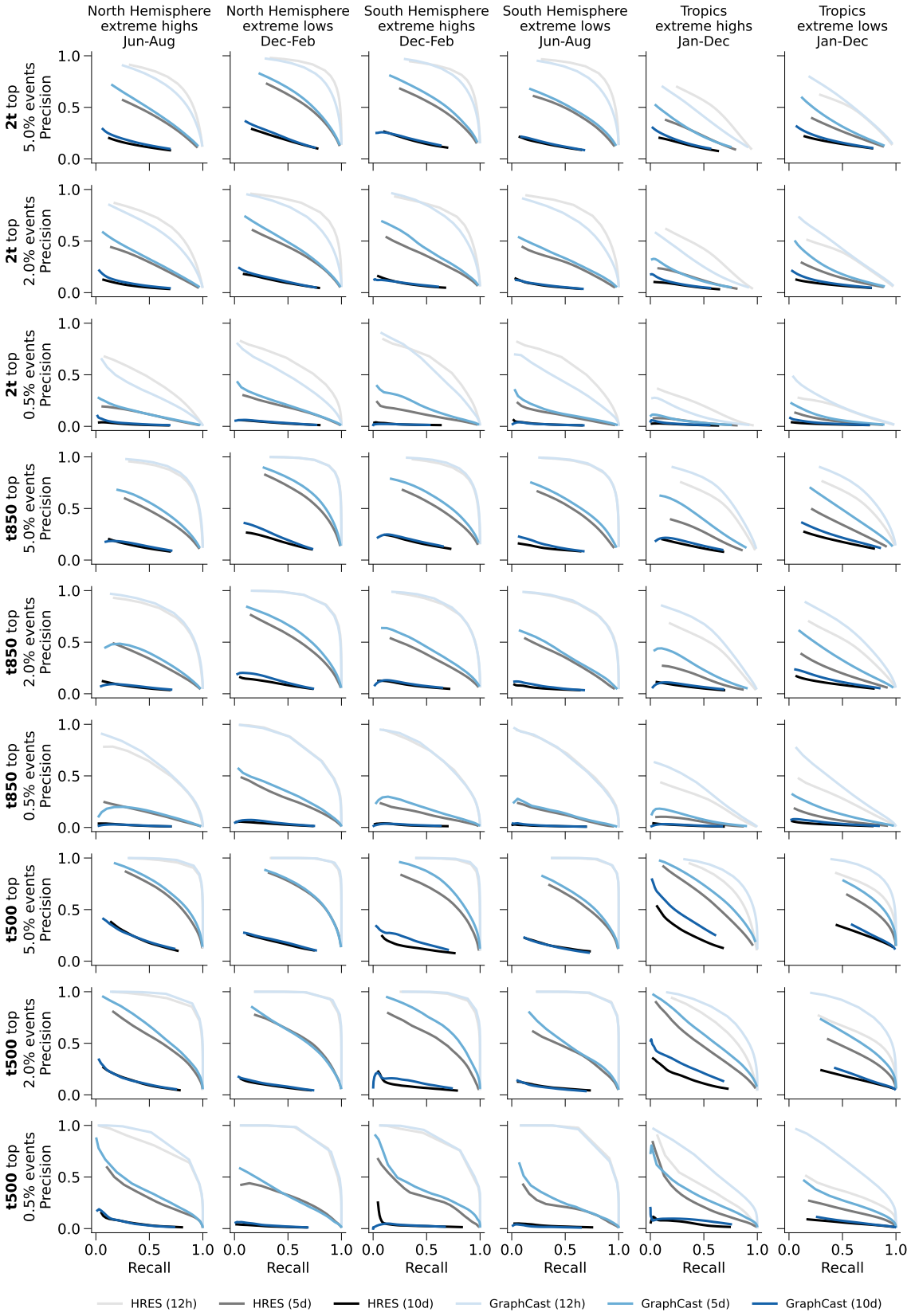

Fig. S45: **Detailed extremes evaluation.** Higher precision and recall is better.

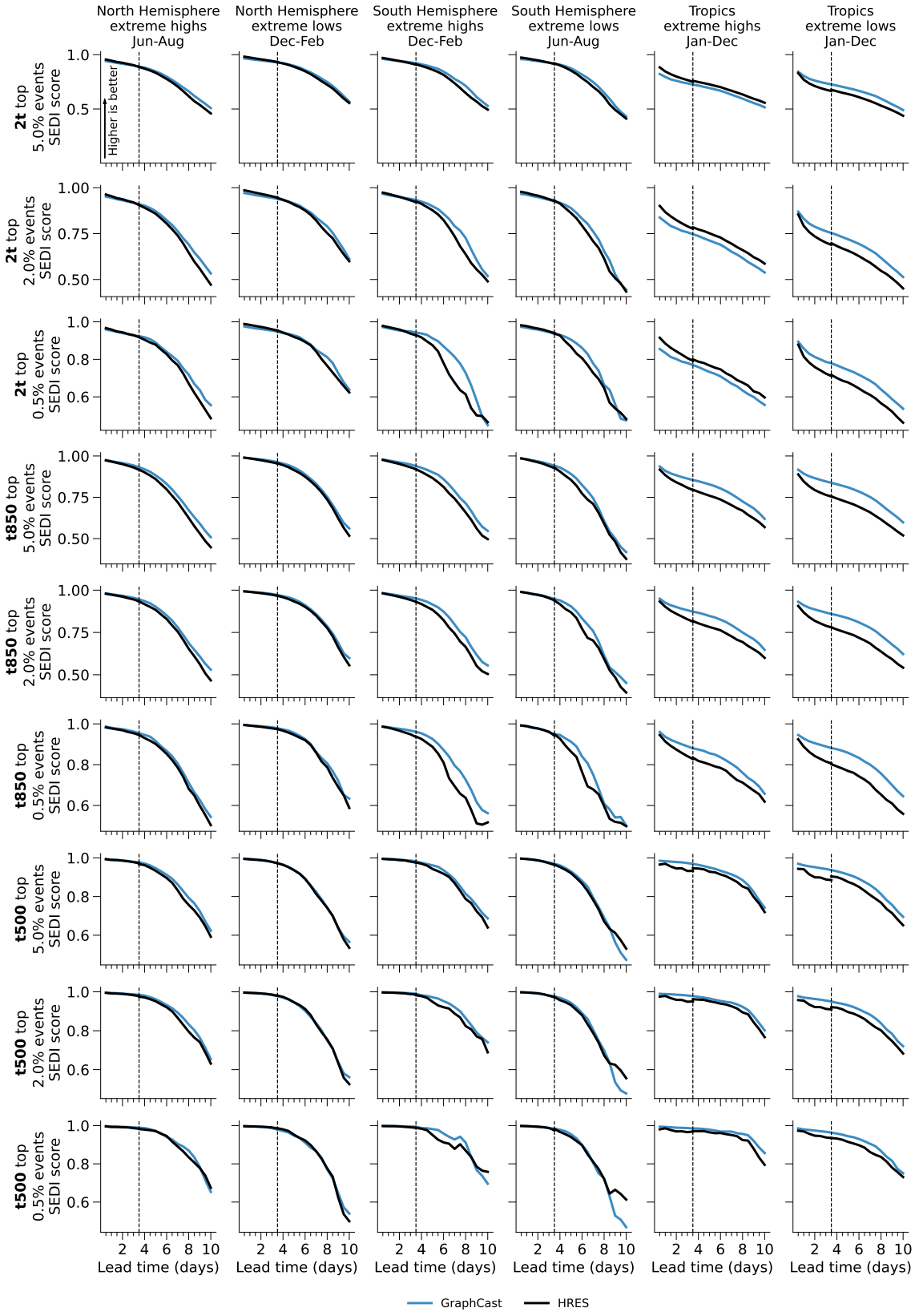

Fig. S46: **Extremes SEDI scores.** Maximum SEDI scores across the extreme prediction precision-recall curves (Figure S45) as function of lead time.

9 Forecast visualizations

In this final section, we provide a few visualization examples of the predictions made by GraphCast for variables 2T (Figure S47), 10U (Figure S48), MSL (Figure S49), Z500 (Figure S50), T850 (Figure S51), V500 (Figure S52), Q700 (Figure S53). For each variable, we show a representative prediction from GraphCast by choosing the example with the median performance on 2018.

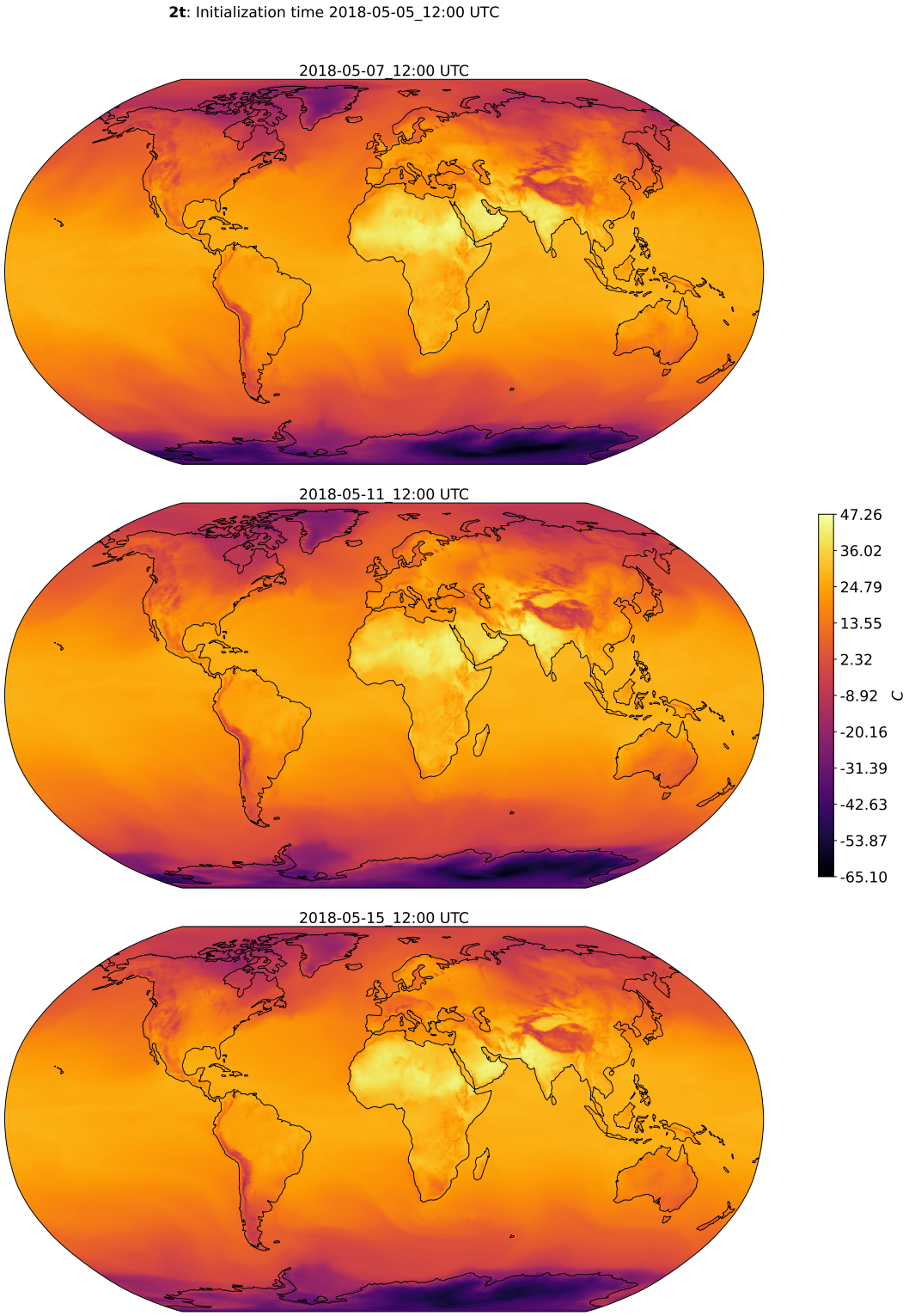

Fig. S47: **GraphCast forecast visualization: 2T.** Forecast initialized at 2018-05-05 12:00 UTC, with plots corresponding to 2, 6, and 10 day lead times.

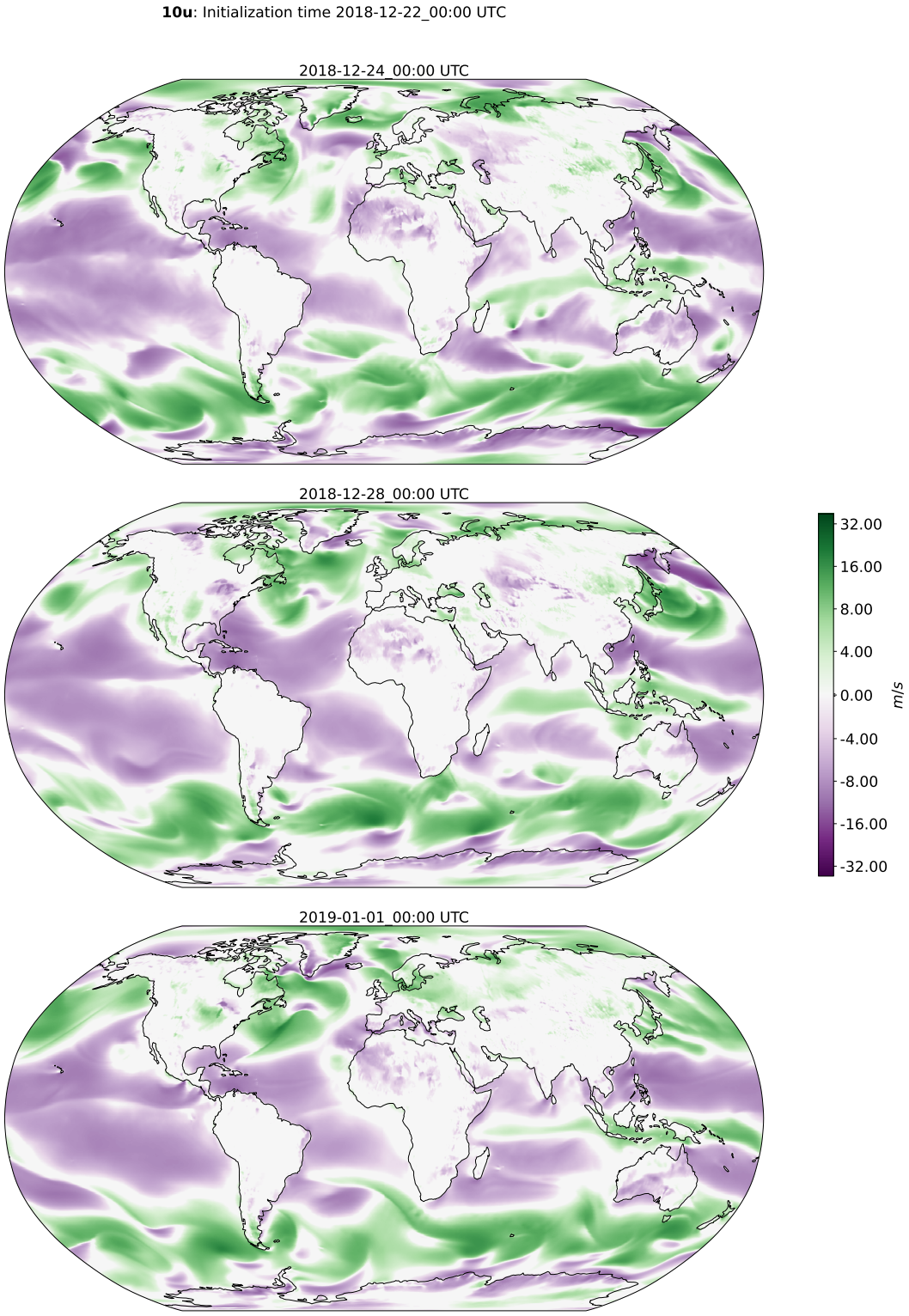

Fig. S48: **GraphCast forecast visualization: 10U.** Forecast initialized at 2018-12-22 00:00 UTC, with plots corresponding to 2, 6, and 10 day lead times.

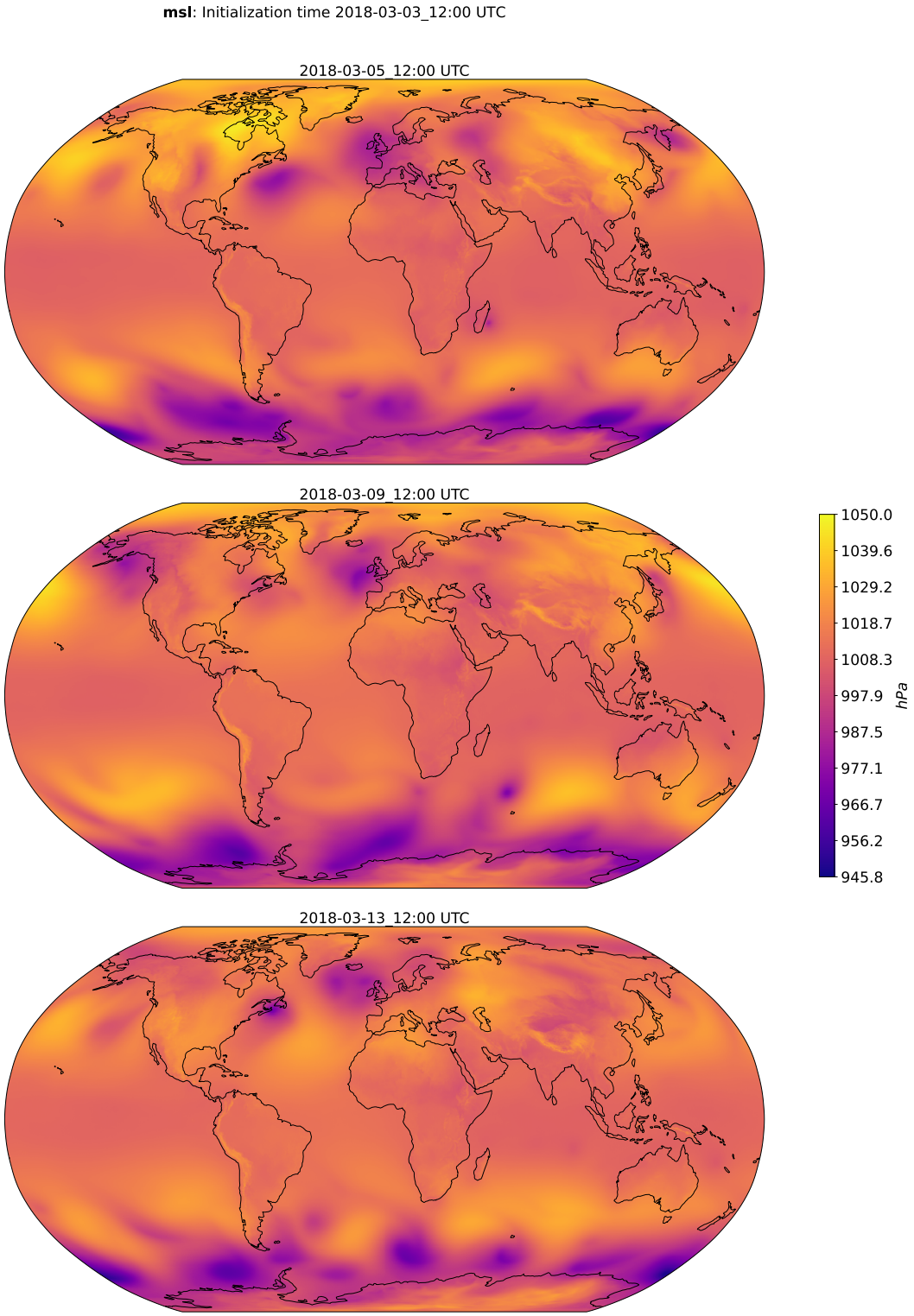

Fig. S49: **GraphCast forecast visualization: MSL.** Forecast initialized at 2018-03-03 12:00 UTC, with plots corresponding to 2, 6, and 10 day lead times.

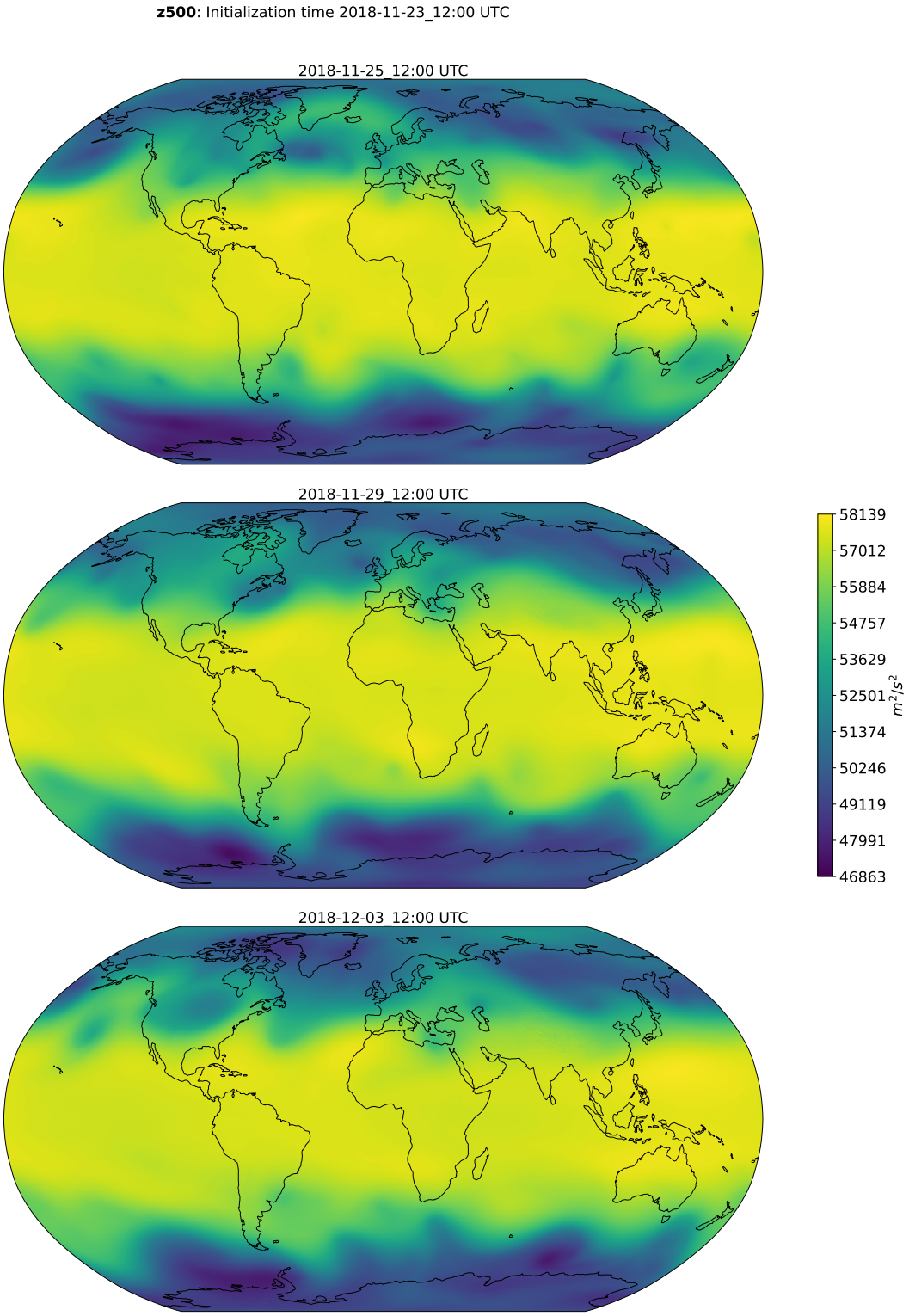

Fig. S50: **GraphCast forecast visualization: z500.** Forecast initialized at 2018-11-23 12:00 UTC, with plots corresponding to 2, 6, and 10 day lead times.

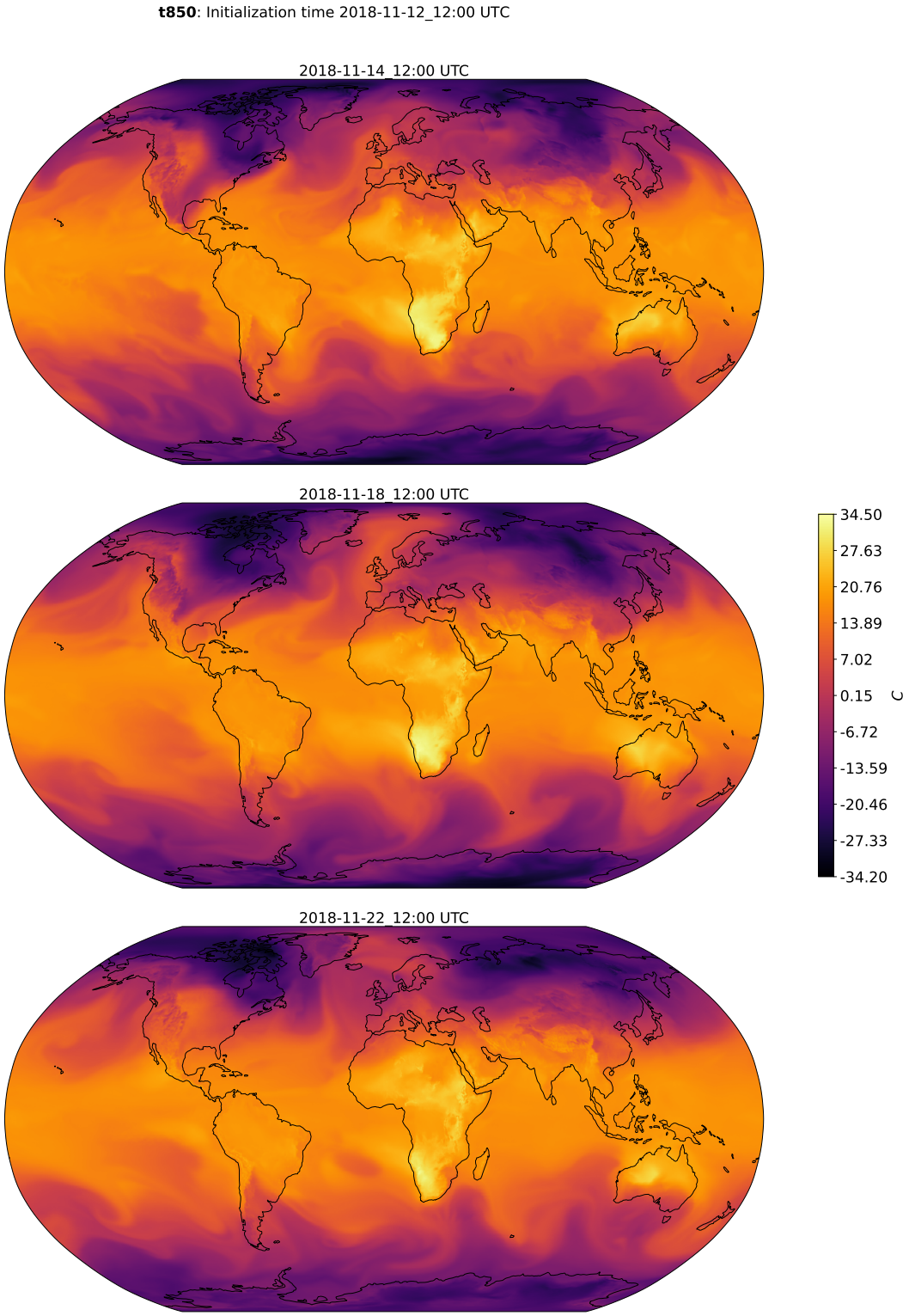

Fig. S51: **GraphCast forecast visualization: T850.** Forecast initialized at 2018-11-12 12:00 UTC, with plots corresponding to 2, 6, and 10 day lead times.

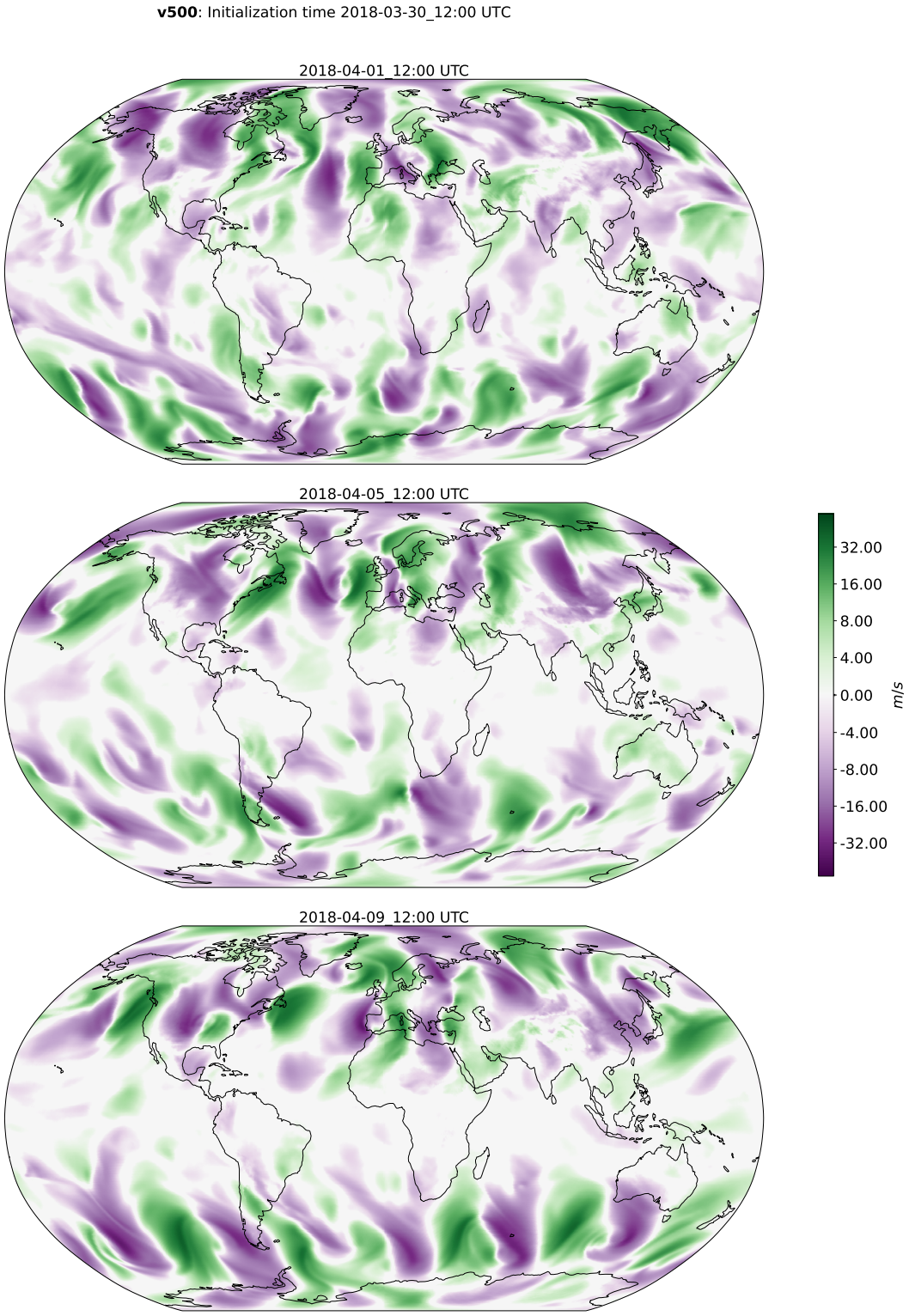

Fig. S52: **GraphCast forecast visualization: v500.** Forecast initialized at 2018-03-30 12:00 UTC, with plots corresponding to 2, 6, and 10 day lead times.

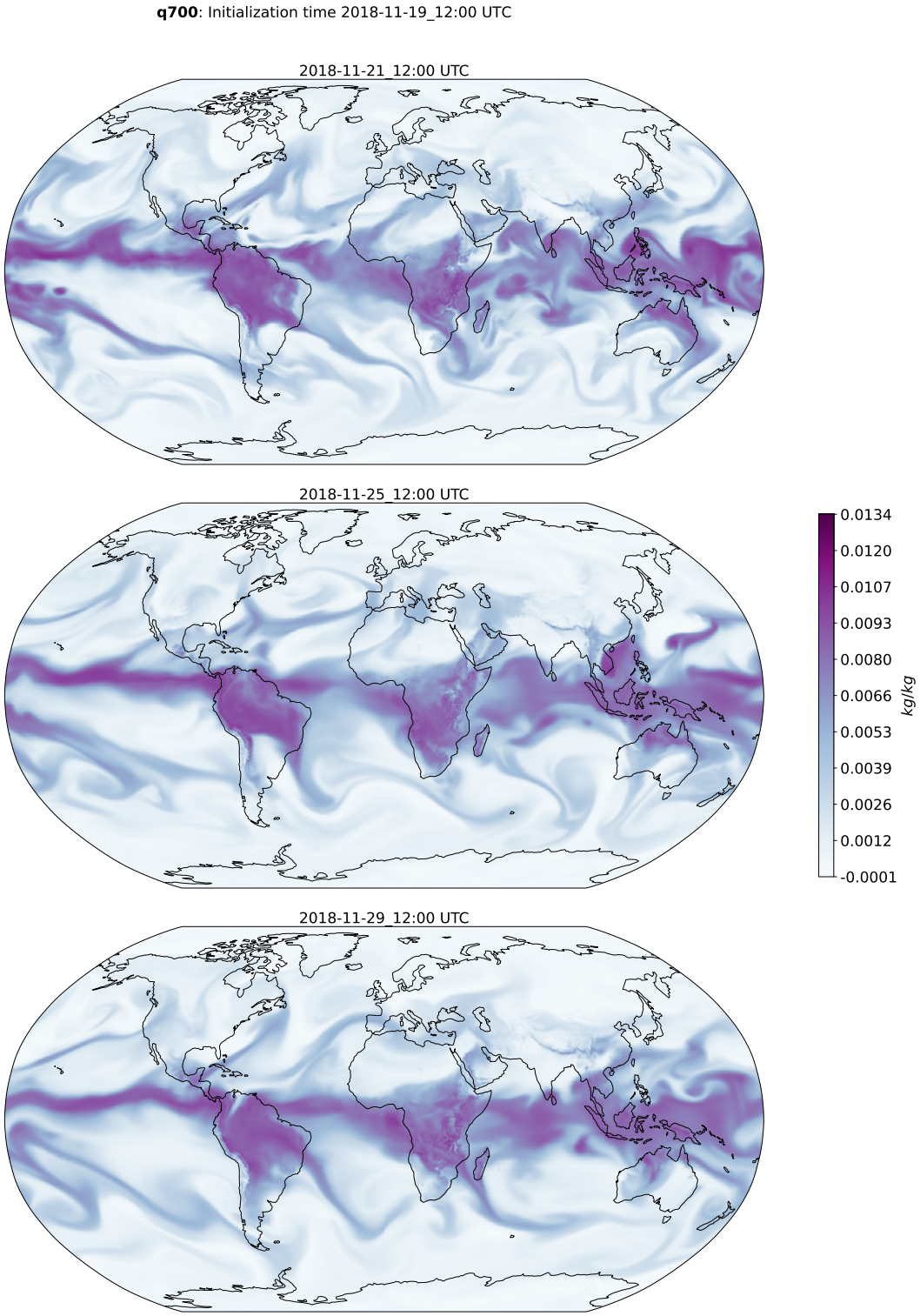

Fig. S53: **GraphCast forecast visualization: Q700.** Forecast initialized at 2018-11-19 12:00 UTC, with plots corresponding to 2, 6, and 10 day lead times.

References

40. S. Malardel, N. Wedi, W. Deconinck, M. Diamantakis, C. Kuehnlein, G. Mozdzynski, M. Hamrud, P. Smolarkiewicz, A new grid for the IFS, https://www.ecmwf.int/node/17262 (2016).

41. D. H. Levinson, H. J. Diamond, K. R. Knapp, M. C. Kruk, E. J. Gibney, Toward a homogenous global tropical cyclone best-track dataset, *Bulletin of the American Meteorological Society* **91**, 377–380 (2010).

42. M. C. Kruk, K. R. Knapp, D. H. Levinson, A technique for combining global tropical cyclone best track data, *Journal of Atmospheric and Oceanic Technology* **27**, 680–692 (2010).

43. A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, I. Polosukhin, Attention is all you need, *Advances in neural information processing systems* **30** (2017).

44. P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Lio, Y. Bengio, Graph attention networks, *arXiv preprint arXiv:1710.10903* (2017).

45. M. Fortunato, T. Pfaff, P. Wirnsberger, A. Pritzel, P. Battaglia, Multiscale meshgraphnets, *arXiv preprint arXiv:2210.00612* (2022).

46. P. Battaglia, R. Pascanu, M. Lai, D. Jimenez Rezende, *et al.*, Interaction networks for learning about objects, relations and physics, *Advances in neural information processing systems* **29** (2016).

47. K. R. Allen, Y. Rubanova, T. Lopez-Guevara, W. Whitney, A. Sanchez-Gonzalez, P. Battaglia, T. Pfaff, Learning rigid dynamics with face interaction graph networks, *arXiv preprint arXiv:2212.03574* (2022).

48. P. Ramachandran, B. Zoph, Q. V. Le, Searching for activation functions, *arXiv preprint arXiv:1710.05941* (2017).

49. J. L. Ba, J. R. Kiros, G. E. Hinton, Layer normalization, *arXiv* (2016).

50. B. Bell, H. Hersbach, A. Simmons, P. Berrisford, P. Dahlgren, A. Horányi, J. Muñoz-Sabater, J. Nicolas, R. Radu, D. Schepers, C. Soci, S. Villaume, J.-R. Bidlot, L. Haimberger, J. Woollen, C. Buontempo, J.-N. Thépaut, The era5 global reanalysis: Preliminary extension to 1950, *Quarterly Journal of the Royal Meteorological Society* **147**, 4186-4227 (2021).

51. I. Loshchilov, F. Hutter, Decoupled weight decay regularization, *arXiv preprint arXiv:1711.05101* (2017).

52. D. P. Kingma, J. Ba, Adam: A method for stochastic optimization, *arXiv preprint arXiv:1412.6980* (2014).

53. I. Loshchilov, F. Hutter, SGDR: stochastic gradient descent with restarts, *CoRR* **abs/1608.03983** (2016).

54. T. Chen, B. Xu, C. Zhang, C. Guestrin, Training deep nets with sublinear memory cost, *arXiv preprint arXiv:1604.06174* (2016).

55. J. Bradbury, R. Frostig, P. Hawkins, M. J. Johnson, C. Leary, D. Maclaurin, G. Necula, A. Paszke, J. VanderPlas, S. Wanderman-Milne, Q. Zhang, JAX: composable transformations of Python+NumPy programs, http://github.com/google/jax (2018).

56. T. Hennigan, T. Cai, T. Norman, I. Babuschkin, Haiku: Sonnet for JAX, http://github.com/deepmind/dm-haiku (2020).

57. J. Godwin, T. Keck, P. Battaglia, V. Bapst, T. Kipf, Y. Li, K. Stachenfeld, P. Veličković, A. Sanchez-Gonzalez, Jraph: A library for graph neural networks in JAX., http://github.com/deepmind/jraph (2020).

58. I. Babuschkin, K. Baumli, A. Bell, S. Bhupatiraju, J. Bruce, P. Buchlovsky, D. Budden, T. Cai, A. Clark, I. Danihelka, C. Fantacci, J. Godwin, C. Jones, R. Hemsley, T. Hennigan, M. Hessel, S. Hou, S. Kapturowski, T. Keck, I. Kemaev, M. King, M. Kunesch, L. Martens, H. Merzic, V. Mikulik, T. Norman, J. Quan, G. Papamakarios, R. Ring, F. Ruiz, A. Sanchez, R. Schneider, E. Sezener, S. Spencer, S. Srinivasan, L. Wang, W. Stokowiec, F. Viola, The DeepMind JAX Ecosystem, http://github.com/deepmind (2020).

59. S. Hoyer, J. Hamman, xarray: N-D labeled arrays and datasets in Python, *Journal of Open Research Software* **5** (2017).

60. R. Swinbank, M. Kyouda, P. Buchanan, L. Froude, T. M. Hamill, T. D. Hewson, J. H. Keller, M. Matsueda, J. Methven, F. Pappenberger, *et al.*, The TIGGE project and its achievements, *Bulletin of the American Meteorological Society* **97**, 49–67 (2016).

61. ECMWF, IFS documentation CY45R1 - Part II : Data assimilation, https://www.ecmwf.int/en/elibrary/80893-ifs-documentation-cy45r1-part-ii-data-assimilation (2018).

62. J. R. Driscoll, D. M. Healy, Computing fourier transforms and convolutions on the 2-sphere, *Adv. Appl. Math.* **15**, 202–250 (1994).

63. A. J. Geer, Significance of changes in medium-range forecast scores, *Tellus A: Dynamic Meteorology and Oceanography* **68**, 30229 (2016).

64. T. Haiden, M. Janousek, J.-R. Bidlot, R. Buizza, L. Ferranti, F. Prates, F. Vitart, Evaluation of ECMWF forecasts, including the 2018 upgrade, https://www.ecmwf.int/node/18746 (2018).

65. T. Haiden, M. Janousek, F. Vitart, L. Ferranti, F. Prates, Evaluation of ECMWF forecasts, including the 2019 upgrade, https://www.ecmwf.int/node/19277 (2019).

66. T. Haiden, M. Janousek, F. Vitart, Z. Ben-Bouallegue, L. Ferranti, C. Prates, D. Richardson, Evaluation of ECMWF forecasts, including the 2020 upgrade, https://www.ecmwf.int/node/19879 (2021).

67. T. Haiden, M. Janousek, F. Vitart, Z. Ben-Bouallegue, L. Ferranti, F. Prates, Evaluation of ECMWF forecasts, including the 2021 upgrade, https://www.ecmwf.int/node/20142 (2021).

68. T. Haiden, M. Janousek, F. Vitart, Z. Ben-Bouallegue, L. Ferranti, F. Prates, D. Richardson, Evaluation of ECMWF forecasts, including the 2021 upgrade, https://www.ecmwf.int/node/20469 (2022).

69. M. J. Rodwell, D. S. Richardson, T. D. Hewson, T. Haiden, A new equitable score suitable for verifying precipitation in numerical weather prediction, *Quarterly Journal of the Royal Meteorological Society* **136**, 1344–1363 (2010).

70. T. Haiden, M. J. Rodwell, D. S. Richardson, A. Okagaki, T. Robinson, T. Hewson, Intercomparison of global model precipitation forecast skill in 2010/11 using the seeps score, *Monthly Weather Review* **140**, 2720–2733 (2012).

71. R. North, M. Trueman, M. Mittermaier, M. J. Rodwell, An assessment of the seeps and sedi metrics for the verification of 6 h forecast precipitation accumulations, *Meteorological Applications* **20**, 164–175 (2013).

72. B. D. Santer, R. Sausen, T. M. L. Wigley, J. S. Boyle, K. AchutaRao, C. Doutriaux, J. E. Hansen, G. A. Meehl, E. Roeckner, R. Ruedy, G. Schmidt, K. E. Taylor, Behavior of tropopause height and atmospheric temperature in models, reanalyses, and observations: Decadal changes, *Journal of Geophysical Research: Atmospheres* **108**, ACL 1-1-ACL 1-22 (2003).

73. ECMWF, IFS documentation CY41R2 - part III: Dynamics and numerical procedures, https://www.ecmwf.int/node/16647 (2016).

74. B. Devaraju, Understanding filtering on the sphere: Experiences from filtering GRACE data, Ph.D. thesis, University of Stuttgart (2015).

75. H. T. Taylor, B. Ward, M. Willis, W. Zaleski, The Saffir-Simpson hurricane wind scale, *Atmospheric Administration: Washington, DC, USA* (2010).