
Supplementary information

A foundation model for the Earth system

In the format provided by the
authors and unedited

Supplementary Information:

A Foundation Model for the Earth System

Table of contents

A	Notation and problem statement	3
B	The Aurora model	3
B.1	3D Perceiver encoder	3
B.2	Multi-scale 3D Swin Transformer U-Net backbone	4
B.3	3D Perceiver decoder	5
B.4	Position, scale, level, and time encodings	5
B.5	Data normalisation	6
B.6	Extensions for 0.1° weather forecasting	6
B.7	Extensions for air pollution forecasting	7
B.8	Extensions for wave forecasting	8
B.9	Model hyperparameters and configurations	9
C	Datasets	10
C.1	Types of datasets	10
C.2	Dataset inventory	10
C.3	Motivation for CMIP6 dataset selection	11
C.4	Training, validation, and test splits	16
C.5	Dataset-specific postprocessing	16
D	Training methods	16
D.1	Training objective	17
D.2	Pretraining methods	18
D.3	Short lead-time fine-tuning	18
D.4	Roll-out fine-tuning	19
E	Data infrastructure	20
F	Verification metrics	21
F.1	Main evaluation metrics	21
F.2	Extreme weather metrics	22
G	Data diversity and model scaling improve atmospheric forecasting	23
G.1	Effects of data diversity	24
G.2	Model scaling laws for surface-level variables	25
H	Comparison against GraphCast, Pangu, and IFS HRES at 0.25° resolution	28
H.1	Experimental setup	28
H.2	Aurora versus GraphCast at 0.25°	28
H.3	Aurora versus Pangu at 0.25°	35
H.4	Aurora versus IFS HRES at 0.25°	35
H.5	Verification against weather station measurements at 0.25°	35
H.6	Extreme event prediction at 0.25°	36
I	Additional results	39

I.1	Comparison against operational CAMS	39
I.2	Comparison against operational IFS HRES-WAM	46
I.3	Details for tropical cyclone forecasting	50
I.4	Comparison against operational IFS HRES at 0.1°	52
I.5	Comparison against weather station measurements	56
I.6	Benefits of modelling at 0.1° resolution	56
I.7	Storm Ciarán	57
I.8	Power spectra	58

A Notation and problem statement

We denote the observed state of the atmosphere (including the surface) at a certain time t via a tensor X^t with dimensions $V \times H \times W$, where V denotes the total number of variables, H specifies the number of latitude coordinates (i.e., the height) and W denotes the number of longitude coordinates (i.e., the width). Thus, we use the $X_{v,i,j}^t$ indexing scheme to refer to the state of variable v at time t and latitude-longitude coordinates given by (i, j) . It is sometimes convenient to split this observed state X^t into its surface (S^t) and atmospheric components (A^t). The surface state S^t is a $V_S \times H \times W$ tensor, where V_S denotes the number of surface variables. The atmospheric state is a $V_A \times C \times H \times W$ tensor where V_A denotes the number of atmospheric variables and C represents the number of pressure levels throughout the atmosphere. Thus, the total number of variables is given by $V = V_S + V_A \times C$.

Given a system state at time t , our goal is to learn to predict the next state at a time $t' > t$. For simplicity, we operate with discrete time units of Δt (which is sufficient for any practical purposes), and consider $t' = t + k\Delta t$ for some positive integer k . To predict the state at the next time step, we learn a simulator $\Phi: (X^{t-\Delta t}, X^t) \mapsto \hat{X}^{t+\Delta t}$, which maps the observed state of the world at the previous time and the current time to a predicted state $\hat{X}^{t+\Delta t}$ at future time $t + \Delta t$. Since Δt might vary from one task to another, we will often simplify this notation to $\Phi: (X^{t-1}, X^t) \mapsto \hat{X}^{t+1}$.

To generate predictions for later time increments, the predictions of the learned simulator can be stacked together in an autoregressive manner. Assuming $\hat{X}^t = X^t$, $\hat{X}^{t-1} = X^{t-1}$, this can be described recursively as

$$\hat{X}^{t+k} = \Phi(\hat{X}^{t+k-2}, \hat{X}^{t+k-1}). \quad (\text{A1})$$

We call this an autoregressive roll-out. It is obtained by applying the function Φ iteratively, a total of k times. Finally, for a certain prediction $\hat{X}^{t'}$, we denote the surface and atmospheric components of that prediction by $\hat{S}^{t'}$ and $\hat{A}^{t'}$, respectively.

B The Aurora model

In this section, we provide a more detailed description of each component of the Aurora model architecture.

B.1 3D Perceiver encoder

Compared to textual data, training large models on a diverse collection of weather datasets is a substantially more challenging task due to the heterogeneous nature of the data. While language is generally homogeneous, weather datasets include different variables, pressure levels and resolutions. Moreover, due to the costs of storing the complete outputs of a simulation, many sources only make subsets of the total data available by reducing the dataset size along one or more of the axes mentioned above. To accommodate such heterogeneous datasets, we design a flexible encoder that maps datasets with varying structure into a standardised 3D tensor that enters the model backbone.

Inputs. The encoder treats all the variables as $H \times W$ images on a regular latitude-longitude grid. For each variable, we include the state at the current time t and the state at time $t - 1$. This results in a $T \times H \times W$ tensor, where $T = 2$ is the time dimension. For a dataset with C pressure levels and V_A atmospheric variables, the state of the atmosphere is represented as a $V_A \times C \times T \times H \times W$. Similarly, for a dataset with V_S surface variables, the state of the surface level is represented as a $V_S \times T \times H \times W$ tensor. In practice, all computations are batched, which adds an additional batch dimension in front of these tensors, which we omit for simplicity.

Static variables. Aurora includes three static variables, which stay constant over time: (1) geopotential at the surface (Z), which provides information about local orography, (2) a land-sea mask (LSM), and (3) soil-type (SLT). Internally in the model, the static variables are incorporated by appending them to the collection of surface-level variables. The static variables are part of ERA5’s invariant surface-level variables.

Level embeddings. As in standard ViTs, we split each of the $H \times W$ images in $P \times P$ patches. The patches at each level are then mapped into a vector in $\mathbb{R}^D$ by a linear layer, i.e. $C \times V_A \times T \times P \times P \mapsto C \times D$ and $V_S \times T \times P \times P \mapsto 1 \times D$. In order to accommodate datasets with different variables, this linear transformation is constructed dynamically for each variable v using a set of weights W_v specific to that variable. Each of the level embeddings is then tagged with an additive level encoding. For the atmospheric

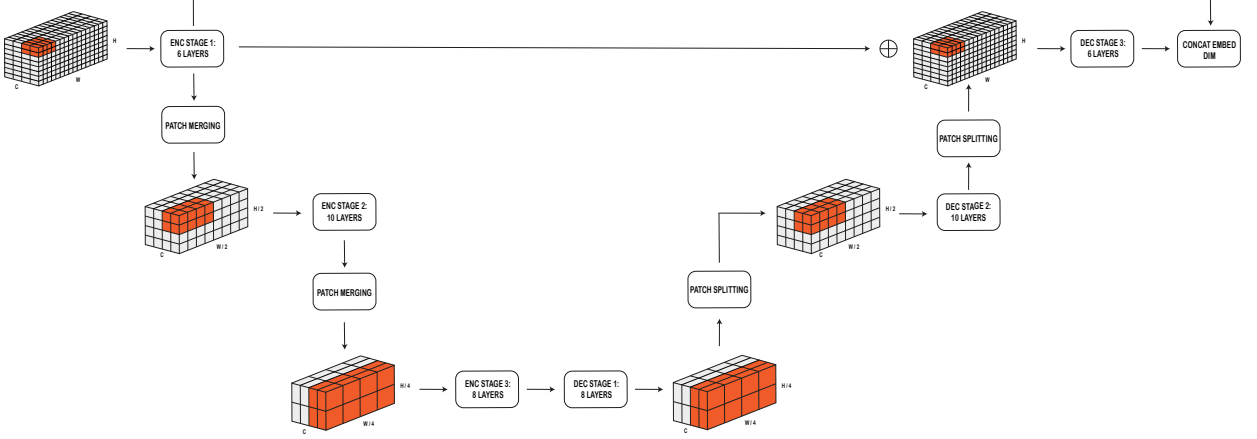


Fig. B1: Overview of the 3D Swin Transformer U-Net backbone of Aurora. The U-Net is formed by an encoder (*left*) and a decoder (*right*). Each of these is composed of three stages with (6, 10, 8) and (8, 10, 6) Swin 3D Transformer^[19] layers, respectively. In total, the backbone has 48 layers. The highlighted tokens show a single 3D self-attention window of size (2, 4, 2) along the depth, width and height dimensions, respectively. The patch merging layers halve the spatial resolution of the 3D tensor, while the patch splitting layers double it. This allows the backbone to simulate the underlying physics at multiple scales. Although not shown in the figure, the embedding dimension of each token is doubled after each patch merging layer and halved after each patch splitting layer.

levels, the level encoding is computed via a sine/cosine encoding of the atmospheric pressure associated with each level (e.g., 150 hPa). For the surface level, we use a fully-learned vector of dimension D .

Level aggregation. The next step is to reduce the number of physical pressure levels in the atmosphere, which can vary across datasets, to a fixed set of latent pressure levels.^[80] introduced a simple aggregation mechanism using attention pooling. Here, we adopt a more expressive scheme that leverages Perceiver modules^[21] consisting of one cross-attention layer followed by a residual MLP. The inputs to the Perceiver are $C_L = 3$ latent query vectors, along with C keys and values vectors computed from the level embeddings via a linear transform. The output of the Perceiver is a $C_L \times D$ tensor, encoding the latent state of the atmosphere. Concurrently, the surface level embedding is simply passed through a residual MLP. This latent state of the surface is then concatenated with the latent state of the atmosphere across the vertical dimension, which yields a $(C_L + 1) \times D$ latent representation of the weather state at the location of each patch.

Scale, position and absolute time embeddings. By gathering the latent representations at each patch, one obtains a 3D tensor $C_L \times H/P \times W/P$, which can be seen as a latent grid on which the backbone will perform the simulation. Therefore, we equip these tokens with information about their latitude and longitude coordinates as well as their physical size (i.e., km^2 per patch – which differs depending on the latitude). Finally, we include absolute time information for each of the patches. For more information on how these encodings are computed, see Supplementary B.4.

B.2 Multi-scale 3D Swin Transformer U-Net backbone

If the 3D tensor that the backbone receives as input can be viewed as a latent 3D mesh on which the simulation is performed, then the backbone can be viewed as a (neural) simulator. Transformers, due to their proven scaling properties^[81] and connections with numerical integration^[54], are a natural architectural choice for this simulation engine. To this end, we opt for a 3D Swin Transformer U-Net^{[19] [50]} architecture (Figure B1), which has also been successfully employed by^[2].

The backbone itself is also composed of an encoder and a decoder, each made out of three stages. After each stage of the encoder, the spatial resolution of the 3D tensor is halved. After each decoder stage, the resolution is doubled and the output is combined with the corresponding outputs of the encoder. This structure enables the backbone to simulate the underlying physics at multiple scales.

Each layer of the backbone is a 3D Swin Transformer layer performing local self-attention operations between tokens in the same regions (called windows), which can be seen as a form of message passing^[58]. One such window (of size (2, 4, 2)) is highlighted in Figure B1. In every other layer, all the windows shift across all

dimensions by half of the window size along that dimension. In this case, the window would shift by (1, 2, 1) along the three axes, which allows the model to propagate information between neighbouring regions. Due to the spherical topology of the Earth, when the window is shifted, we allow the left and right side of the images to communicate directly as in Bi et al. [2]. Overall, this procedure emulates the local computations performed by numerical integration methods, while avoiding the quadratic complexity of vanilla Transformers, which is impractical for the resolutions Aurora operates at. In practice, we use a window size of (2, 12, 6) for all of our experiments, similarly to Bi et al. [2].

To increase stability throughout training, we use the res-post-norm layer normalisation introduced in Liu et al. [50], but opt for a standard dot-product attention operation instead of the cosine attention procedure from Swin v2 [50]. To obtain a simple and flexible backbone that can operate at multiple resolutions, we do not use any form of positional bias in the attention operations as previously done by Bi et al. [2], Liu et al. [19] Liu et al. [50], which requires inputs with a fixed spatial resolution. Instead, we opt for positional and scale encodings in the encoder, as described in the previous section, which does not impose any constraints on the inputs.

Finally, we note that the encoding procedure we employ allows us to flexibly use a small number of latent levels, which in turn frees up memory for massively scaling the number of parameters in the backbone. Compared to the backbone in Bi et al. [2], which has 16 layers and two stages, our backbone has an additional stage and contains 48 layers.

B.3 3D Perceiver decoder

The decoder maps the standardised outputs of the latent simulation back into images on the regular lat-lon grid. Its structure mirrors that of the encoder. The three latent atmospheric pressure levels are de-aggregated into C , D -dimensional embeddings (i.e., one per output atmospheric level). As in the encoder, this is done via a Perceiver layer [21], which uses the sine/cosine embeddings of the output levels' pressure as queries. These vectors are then decoded into $P \times P$ patches via a linear layer. The latent surface level is decoded directly. Analogously to the patch embedding layer in the encoder, the linear layer creating the output patches is constructed dynamically by selecting the weights associated with each variable. This overall architecture allows the decoder to output predictions at arbitrary pressure levels, for an arbitrary set of variables.

B.4 Position, scale, level, and time encodings

Given the minimum and maximum wavelengths $\lambda_{\min}$ and $\lambda_{\max}$ we want to capture, we use a Fourier encoding of the following form to encode a value x into a D -dimensional vector,

$$\text{Emb}(x) = \left[\cos \frac{2\pi x}{\lambda_i}, \sin \frac{2\pi x}{\lambda_i} \right] \text{ for } 0 \leq i < D/2, \quad (\text{B2})$$

where the f_i are log-spaced values between the minimum and maximum wavelength, i.e.,

$$\lambda_i = \exp \left(\log \lambda_{\min} + i \cdot \frac{\log \lambda_{\max} - \log \lambda_{\min}}{D/2 - 1} \right). \quad (\text{B3})$$

In the following, we describe how the value x is computed for all the encodings and how $\lambda_{\min}$ and $\lambda_{\max}$ are chosen.

Positional encoding. A two-dimensional positional encoding is used to account for the spatial position of each token that enters the Transformer backbone. For an input token of dimension D , the first $D/2$ dimensions are used for latitude information while the next $D/2$ dimensions for longitude information. In contrast to standard positional encodings used in ViT [20] [80], where the coordinate IDs of the patch are used, we use a geometry-aware positional encoding given by the average latitude and longitude of each patch. We set $\lambda_{\min} = 0.01$ and $\lambda_{\max} = 720$.

Scale encoding. A separate, non-trainable, two-dimensional positional encoding is added to account for the physical scale of each patch token. Specifically, give a patch with minimum and maximum latitudes

ϕ_1, ϕ_2 , and minimum and maximum longitudes θ_1, θ_2 (measured in radians), we compute the area of the patch using the formula

$$A = R^2(\sin \phi_2 - \sin \phi_1)(\theta_2 - \theta_1), \quad (\text{B4})$$

where $R = 6371$ km is the radius of the Earth. Therefore, patches that originate from a high-resolution input dataset (e.g., IFS HRES data) will yield small areas, while patches that originate from coarser datasets (e.g., CAMS) will be assigned encodings with larger magnitude. We set $\lambda_{\min}$ to a sufficiently small value and $\lambda_{\max}$ to the surface area of the entire Earth.

Pressure level encoding. Here, we simply encode the value of the atmospheric pressure associated with each level (e.g., 500 hPa). We set $\lambda_{\min} = 0.01$ and $\lambda_{\max} = 10,000$.

Absolute time encoding. We encode the absolute time associated with a certain input as the number of hours since January 1, 1970. We set $\lambda_{\min}$ to one hour and $\lambda_{\max}$ to the number of hours in a year. Therefore, this encoding is able to capture important information such as time of day, time of the week, month of the year, etc.

B.5 Data normalisation

Aurora normalises all variables before processing them in the encoder and unnormalises the output of the decoder to produce the final predictions. Every surface-level variable and every pressure level of every atmospheric variable is normalised separately by a spatially constant scale and centre:

$$X_{v,i,j,\text{normalised}}^t = \frac{X_{v,i,j}^t - \text{centre}_v}{\text{scale}_v}. \quad (\text{B5})$$

We collectively call the scales and centres the normalisation statistics. The centres are estimated by the empirical means computed over the whole ERA5 training data, and the scales are estimated by the empirical standard deviations computed over the whole ERA5 training data. These normalisation statistics are then used for all datasets. Final predictions are produced by unnormalising the output of the decoder:

$$\hat{X}_{v,i,j}^t = \text{scale}_v \cdot \hat{X}_{v,i,j,\text{normalised}}^t + \text{centre}_v, \quad (\text{B6})$$

where $\hat{X}_{v,i,j,\text{normalised}}^t = \Phi(X_{v,i,j,\text{normalised}}^{t-1}, X_{v,i,j,\text{normalised}}^{t-2})$ represents the raw, unnormalised output of the decoder.

B.6 Extensions for 0.1° weather forecasting

Due to the increase in resolution, it becomes extremely challenging to make a large vision model like Aurora fit in the available GPU memory, which is why we perform a few minor modifications to the architecture at fine-tuning time.

Increase patch size. We increase patch size by a factor of $2.5\times$, from 4 (used in pretraining) to 10. Since this corresponds to the increase in resolution from 0.25° to 0.1° , the spatial resolution of the backbone input remains the same as in pretraining. Therefore, there is no memory increase in the backbone computations, where most of the parameters reside. To ensure maximum transfer of skills from pretraining to fine-tuning, we use a bilinear interpolation to interpolate from the smaller patch size to the larger patch size, then we scale the magnitude of the interpolated weights by the ratio of the two patch sizes in order to preserve the magnitude of the inputs^[59]. This ensures that the pretrained model can make reasonable predictions at 0.1° even before it is fine-tuned. We find that this also makes fine-tuning significantly faster.

Remove backbone layers. We remove two layers from the two middle stages of the backbone encoder and decoder, respectively. Thus, the number of layers for all stages becomes (6, 8, 8) for the backbone encoder and (8, 8, 6) for the backbone decoder. Since the model is pretrained using stochastic depth, the model is robust to this change and we do not notice any significant drop in the initial loss before fine-tuning.

B.7 Extensions for air pollution forecasting

In Section 3, we fine-tune Aurora to predict concentrations of air pollutants in addition to a standard collection of meteorological variables. CAMS uses spatially and temporally disaggregated emissions data as inputs. These inputs allow natural factors (i.e., wildfires, vegetation, etc.) and anthropogenic factors (i.e., vehicle combustion, energy production, etc.) to influence the levels of the air pollutants simulated by CAMS. Aurora does not use any emissions data as input, except for some additional static variables which are fixed across all times and experiments. Instead, Aurora must implicitly learn to account for these effects, but likely would improve if direct access to CAMS emissions factors, or other similar emissions data, were available. We therefore slightly adapt Aurora to better predict these variables. All adaptations are carefully described in this section.

12-hour model. Some pollutants, such as NO, show clear diurnal cycles (Section 7). We therefore fine-tune a version of Aurora that was pretrained with time step equal to 12 hours instead of 6 hours. A 12-hour model which takes in two previous time steps can use the state at time $t - 24$ h to make predictions for time t . The 12-hour model was pretrained exactly like the 6-hour model, but only for 80.5 k steps, corresponding to roughly a week and a half of training.

Differencing. Since some pollutants show clear diurnal cycles, we build the ability to predict the difference with respect to a previous state explicitly into the model. Only for the new pollution variables, Aurora explicitly predicts the difference with respect to either time $t - 12$ h or time $t - 24$ h. Variables NO, TC NO, NO₂, TC NO₂, SO₂, TC SO₂, PM₁, PM_{2.5}, and PM₁₀ show clear diurnal cycles, so for these variables we predict the difference with respect to time $t - 24$ h. For variables CO, TC CO, O₃, and TC O₃ the behaviour is less clear, so for these variables we predict the difference with respect to time $t - 12$ h.

We do not restrict Aurora to necessarily predict the difference. In the decoder, we introduce a modulation factor, initialised to one, produced by an additional head:

$$\hat{X}^t = \underbrace{[\Phi_{\text{mod}}(X^{t-1}, X^{t-2}) + 1]}_{\text{modulation factor}} X^a + \underbrace{\Phi_{\text{pred}}(X^{t-1}, X^{t-2})}_{\text{delta or direct prediction}}, \quad (\text{B7})$$

where a is either $t - 1$ or $t - 2$, depending on the variable. Φ_{mod} and Φ_{pred} are two different heads of the decoder, both initialised to zero. Therefore, at initialisation, Aurora predicts the appropriate persistence prediction only for the new pollution variables.

Normalisation for concentrations of air pollutants. Fine-tuning on CAMS analysis happens in two stages: an initial fine-tuning stage on CAMS reanalysis data, and the main fine-tuning on CAMS analysis data (see Table C5 and Supplementary D.3). For the initial fine-tuning stage on CAMS reanalysis, normalisation statistics for the new variables are computed over the whole CAMS reanalysis fine-tuning data; and for the main fine-tuning on CAMS analysis, normalisation statistics for the new variables are computed over the whole CAMS analysis training data. Due to the heterogeneous and skewed nature of air pollution variables, the empirical standard deviation is not a good estimate of the scale. Instead, for all air pollution variables v , we set $\text{centre}_v = 0$, and estimate scale_v by half the spatial maximum averaged over time,

$$\text{scale}_v = \frac{1}{2} \cdot \frac{1}{T} \sum_{t=1}^T \max \{X_{v,i,j}^t : i = 1, \dots, H \ j = 1, \dots, W\}. \quad (\text{B8})$$

By construction of this normalisation, the normalised air pollution variables will typically be in the range $[0, 2]$.

Transformation of concentration variables. Variables which are concentration values have a large dynamic range by exhibiting important structure on varying orders of magnitude. Including these variables without any modification makes the network sensitive to high-magnitude values (i.e., spikes) and insensitive to low-magnitude values. One solution is to use the logarithm of these variables. However, including the logarithm of positive variables makes the network overly sensitive to extremely-low-magnitude values (i.e., zero becomes negative infinity). We therefore include a linear combination of the original variable x and

$\log(x)$ which clips away the weaknesses of x and $\log(x)$:

$$x_{\text{transformed}} = c_1 \min(x, 2.5) + c_2 \frac{\log(\max(x, 10^{-4})) - \log(10^{-4})}{-\log(10^{-4})}, \quad (\text{B9})$$

where c_1 and c_2 are initialised to $\frac{1}{2}$. The particular constants in the above transformation assume that $x_{\text{transformed}}$ is normalised to usually be in the range $[0, 2]$. For $10^{-4} \leq x \leq 2.5$, which is the most interesting range, $x_{\text{transformed}}$ depends both on x , which is sensitive to normal-magnitude values, and on $\log(x)$, which is sensitive to low-magnitude values. For $x \geq 2.5$, $\min(x, 2.5)$ clips x to 2.5, so $x_{\text{transformed}}$ then only depends on x through $\log(x)$. For $x \leq 10^{-4}$, the c_2 -term is zero exactly, so $x_{\text{transformed}}$ then only depends on x through x .

Additional static variables. To help Aurora better understand diurnal cycles and human behaviour, we extend the static variables to include the time of day (normalised to $[0, 1)$), the day of week (normalised to $[0, 1)$), and the day of the year (divided by 365.25). These normalised values x are included as two constant masks $\cos(2\pi x)$ and $\sin(2\pi x)$. We also include more pollution-related static variables: the monthly average anthropogenic emissions in the year 2019 summed across all industries for (1) ammonia, (2) CO, (3) the sum of NO and NO₂, and (4) SO₂. These monthly averages x are included in two forms: as

$$\frac{x}{\max(x)}, \quad \text{and} \quad \left[\log \left(\max \left(\frac{x}{\max(x)}, 10^{-10} \right) \right) - \log(10^{-10}) \right] / 10. \quad (\text{B10})$$

The monthly averages can be downloaded from CAMS global emission inventories at <https://ads.atmosphere.copernicus.eu/cdsapp#!/dataset/cams-global-emission-inventories> and are regridded to the resolution of CAMS analysis data. In total, there are $3 \cdot 2 + 4 \cdot 2 = 14$ additional static variables.

Finally, whereas Aurora originally only includes the static variables as additional surface-level variables, we now also include them as additional atmospheric variables by replicating them for every pressure level.

Variable-specific clipping. SO₂ at 850 hPa and nearer to the surface tends to be particularly spikey. To help stabilise roll-outs, during and after LoRA roll-out fine-tuning, we clip the predictions only for SO₂ at 850 hPa and above at 1. This clipping happens before unnormalisation.

Patch size decrease. The lower resolution of CAMS analysis data (0.4° instead of 0.25°) allows us to use a smaller patch size of 3 instead of patch size 4. To ensure maximum transfer, we initialise the size 3 patches following the procedure described in Supplementary B.6.

Pressure-level-specific patch embeddings. Near the surface, air pollution variables are strongly affected by human behaviour. However, near the top of the atmosphere, air pollution variables evolve according to complicated atmospheric chemistry dynamics. To deal with the increased variety of behaviours across pressure levels for the atmospheric variables, we make the patch embeddings in the 3D Perceiver encoder and the patch reconstructions in the 3D Perceiver decoder dependent on the pressure level. The pressure-level-dependent patches are initialised with existing pressure-level-independent patches wherever possible and otherwise with uniform random values on the encoder side and with zeros on the decoder side.

Additional 3D Perceiver decoder. To help Aurora with the difference in behaviour between meteorological and air pollution variables, we use a separate 3D Perceiver decoder for the air pollution variables. This separate 3D Perceiver decoder is initialised to the one learned during pretraining.

32-bit floating point computation. To better handle low-magnitude values, all computations before and after the backbone are performed in `float32`. The backbone operates in `bf16` mixed precision mode.

B.8 Extensions for wave forecasting

In Section 4, we fine-tune Aurora to forecast ocean surface waves. All adaptations of Aurora for wave forecasting are described in this section.

Initialisation for new wave variables. All new wave variables are modelled as surface-level variables. The new patch embeddings are all initialised to zero, except for the patch embeddings of 10UN and 10VN, which are initialised to the patch embeddings of 10U and 10V learned during pretraining.

Table B1: The four Aurora model configurations used for the model scaling experiments in Supplementary G. For the 290 M model, the attention head dimensionality of the first layer of the encoder and last layer of the decoder was 80 instead of 64.

Model size	Backbone encoder layers	Backbone decoder layers	Embed dim.	Attention head dim.	Perceiver heads
117 M	(2, 6, 2)	(2, 6, 2)	256	64	8
290 M	(4, 8, 4)	(4, 8, 4)	320	64	8
660 M	(6, 8, 8)	(8, 8, 6)	384	64	12
1.3 B	(6, 10, 8)	(8, 10, 6)	512	64	16

Angle-valued variables. All angle-valued variables, which are MWD, MDWW, MDTS, MWD1, and MWD2, are transformed with $x \mapsto (\sin(x), \cos(x))$ prior to running the encoder. The model has separate patch embeddings for the sine and cosine components of angle-valued variables. After running the decoder, sine and cosine components are transformed back to angles using $(\sin, \cos) \mapsto \text{atan2}(\sin, \cos)$. The angle-valued variables have normalisation mean zero and normalisation scale one.

Density channels for missing data. Wave variables are undefined above land. In addition, wave variables can be missing above water, for example in the case that there is sea ice or a swell component is just absent. The model therefore must be able to handle data missing in the inputs and predict missing values in the outputs. We accomplish this by, for every wave variable, incorporating a so-called *density channel*^[33].

For a variable, the density channel is one if data are present and zero if data are absent. In the variable, missing values are then simply set to zero after normalisation. We need to include such a density channel, as otherwise the model would not be able to distinguish between a zero in the variable and a missing value. The model treats the variable and the associated density channel as separate surface-level variables with separate patch embeddings. The model also predicts the density channels, which is how the model can predict that a variable is absent at a specific location. To bound the density channels to the range $[0, 1]$, we use the sigmoid function.

When running the model autoregressively, it will take its own predictions for density channels as inputs. In this case, to avoid any distribution mismatch, we set all density channels to one wherever it is greater than $\frac{1}{2}$, and we set all density channels *and* associated data channels to zero wherever the density channels are less than $\frac{1}{2}$.

Additional static variables. We include two additional static variables: the bathymetry, and a mask which is one wherever HRES-WAM models wave variables (between latitudes -78° and 90°) and zero elsewhere. The additional static variables are normalised to the range $[0, 1]$.

Additional layer normalisation. When fine-tuning Aurora to the wave data, to stabilise training, we apply a layer normalisation to the keys of the first level aggregation attention block and another layer normalisation to the queries of the first level aggregation attention block. These layer normalisations are applied before the keys and queries are split across multiple heads.

B.9 Model hyperparameters and configurations

The 1.3B parameter Aurora model which we use in the main experiments instantiates the architecture as follows. The embedding dimension in the encoder and first stage of the backbone is 512. This dimension doubles at every subsequent stage of the backbone. The number of attention heads in the backbone is selected such that the embedding dimension per head is 64 throughout the backbone. Due to the concatenation at the end of the backbone, the embedding dimension in the decoder is 1024. In the Perceiver layers of the encoder and decoder, we use an increased number of cross-attention heads (16), in order to give the model fine-grained control over how the latent state of the atmosphere is constructed.

For the model scaling experiment in section Supplementary G, we instantiate smaller versions of this model by reducing the number of backbone layers, the embedding dimension and the number of (cross) attention heads, while always preserving the attention head dimension of 64 (Table B1).

C Datasets

We use several different datasets throughout our experiments with Aurora. This section provides a detailed inventory of all these datasets and the various details behind them.

C.1 Types of datasets

The datasets used in the paper can be classified into five categories: analysis, reanalysis, forecasts, reforecasts, and climate simulation datasets. Observational data such weather station observations and satellite data only measure the world in very select ways in particular regions, namely where the weather stations and satellites are located. To produce a gridded estimate of the state of the world throughout the entire atmosphere, numerical models are used to fill in the missing gaps. The different kind of datasets differ in precisely which observational data they incorporate and how they use a numerical model.

Reanalysis data is regarded as the best estimate of the state of the weather at a particular point in time. It uses a fixed numerical model to assimilate observations over a longer time window, including information from both the past as well as the future, to make an estimate about the present. As a result, reanalysis is not available in real time since it incorporates information the future, and it therefore cannot be used to initialise operational NWP forecasts.

To initialise operational NWP forecasts, one instead maintains *analysis data*. Analysis data is produced in a recursive manner: just before NWP forecasts are scheduled to be run, the analysis assimilation system combines the most recent observational data with previous NWP forecasts to produce an estimate of the NWP initial condition. Compared to reanalysis data, analysis data is available in real time because it only incorporates observational data from the past.

In addition, one can make after-the-fact *reforecasts* by using reanalysis data as the initial condition. These initial conditions are reanalysis data, so they are not available when making forecasts operationally. Nevertheless, the purpose of reforecasts is to indicate how NWP models would have performed if they had access to the best possible global state at initialisation time.

Finally, *climate simulations* model the physics, chemistry, and biology of the atmosphere to generate potential climate scenarios under different forcing factors (e.g. various emission levels).

C.2 Dataset inventory

We enumerate all the datasets used in the paper below. Unless otherwise specified, we use the 13 pressure levels from WeatherBench2^[60]: 50, 100, 150, 200, 250, 300, 400, 500, 600, 700, 850, 925, and 1000 hPa.

- **ERA5**^[61,62] is a global reanalysis dataset from ECMWF for weather. It is widely regarded as the best estimate we have of the total state of the weather at 0.25°.
- **HRES forecasts**^[63] refers to the high-resolution version of the operational NWP forecasting model run by ECMWF. It is considered to be the most accurate NWP forecasting model and it runs at 0.1° resolution. We use it both at its native resolution and regridded to 0.25°.
- **HRES T0** provides the initial conditions used to initialise the HRES forecasts and it is often considered as the ground truth against which to evaluate the quality of the forecasts^[60].
- **HRES analysis** represents the official analysis product of ECWMF. It contains an additional assimilation step on top of HRES T0. In this sense, it provides a slightly more accurate source of truth.
- **IFS ENS**^[64] is the ensemble version of IFS, with 50 members run at a slightly coarser resolution of 18 km (prior to June 27 2023, after which the resolution increased to 9 km with model cycle 48R1). The 50 ensemble members are generated with perturbed initial conditions and stochastic model physics within the IFS model. We use the dataset from the WeatherBench2 repository^[60], where there are only 3 pressure levels available: 500, 700, and 850 hPa.
- **IFS ENS mean** contains the mean predictions for each variable based on IFS ENS. It is provided by WeatherBench2^[60] and contains the same 3 pressure levels as IFS ENS.

- **GFS forecasts**^[65] provides operational forecasts with a base resolution of 18 km. Here we use the data re-gridded to 0.25°. The zero time (initialisation) of these forecasts is the real-time operational analysis, derived analogously to HRES-T0.
- **GFS T0**^[65] refers to the real-time operational analysis, obtained from the zero time (initialisation) of the GFS forecasts.
- **GEFS reforecasts**^[66] is based on 21 ensemble members to address underlying uncertainties in the input data such limited coverage, instruments or observing systems biases, as well as the limitations of the model itself. In practice, such large quantities of data are only archived for a few preceding months. Therefore, we use the *reforecast* data, spanning 2000-2019, initialised with reanalysis initial conditions at 00 UTC each day. In this setting, there are only five ensemble members, all of which are included. GEFS has 6 pressure levels, we use the 3 that line up with those from WeatherBench2: 850, 925, 1000 hPa.
- **CMIP6** is a climate model inter-comparison project, combining various climate modelling experiments, which include land, sea, atmosphere, and aerosol variables. We have two datasets from CMIP6: CMCC-CM2-VHR4^[67] and ECMWF-IFS-HR^[68], which each have 7 pressure levels: 50, 250, 500, 600, 700, 850, 925 hPa.
- **MERRA-2**^[69] is an atmospheric reanalysis dataset from NASA’S Global modelling and Assimilation Office, incorporating space-based observations of aerosols. We use pressure levels corresponding to the 13 levels from WeatherBench2.
- **HRES-WAM**^[30] refers to analysis and forecast data from ECMWF’s ocean wave model called HRES-WAM. This model is coupled to HRES, operates at 0.1° resolution, and models surface-level variables describing ocean waves. We regrid this data to 0.25° resolution.
- **CAMS**^[70] refers to analysis and forecast data from the Copernicus Atmospheric Monitoring Service. The data has 0.4° resolution and includes meteorological variables as well as variables describing the composition of the atmosphere, such as concentrations of air pollutants. CAMS undergoes frequent model updates. We use the 13 pressure levels from WeatherBench2.
- **CAMS reanalysis**^[27] refers to the fourth generation ECMWF global reanalysis of atmospheric composition (EAC4) from the Copernicus Atmospheric Monitoring Service^[76]. The data has 0.75° resolution and, like CAMS, includes meteorological and variables as well as variables describing the composition of the atmosphere. However, unlike CAMS, CAMS reanalysis is produced with a single, albeit considerably older, IFS model cycle: CY42R1. CAMS reanalysis is considered to be less accurate than recent CAMS data because the model cycle is so much older and the resolution is coarser; a quantitative comparison of CAMS and CAMS reanalysis against station observations can be found at <https://aeroyal.met.no/>. We use the 13 pressure levels from WeatherBench2.

Collectively, these data sources open different windows onto the underlying atmospheric dynamics, and expose Aurora to different factors that reflect variability with respect to initial conditions, model parameterizations, and chaotic dynamics. The range of variables used, along with their definitions, are detailed in Table C2.

In Table C3, Table C4, and Table C5, we detail the dataset details used for pretraining, fine-tuning, and evaluating Aurora, respectively. The following configuration of datasets was used for the final pretraining run: ERA5, 1-day HRES-0.25 forecasts, 1-day GFS forecasts, GFS T0, CMCC-CM2-VHR4, and ECMWF-IFS-HR. This is configuration C4 in the data scaling study (Supplementary G.1), which yielded the overall best results.

C.3 Motivation for CMIP6 dataset selection

We selected the CMCC-CM2-VHR4 and ECMWF-IFS-HR from CMIP6 for several reasons:

1. **Resolution compatibility:** CMCC-CM2-VHR4 provides data at 0.25°, matching ERA5, while ECMWF-IFS-HR at 0.45° offers a complementary intermediate scale which is well matched for the atmospheric chemistry setting. As detailed in Table C4, this resolution diversity helps Aurora learn scale-aware representations while maintaining sufficient fidelity for our target applications.

Table C2: Naming of the variables used from the datasets, including units. Surface-level variables are available as a single value per latitude–longitude point. Atmospheric variables are instead available at a range of levels of constant air pressure throughout the atmosphere, therefore there are multiple values per latitude–longitude point. Atmospheric pollutants are measured as concentrations throughout the layers of the atmosphere with units of kg m^{-3} . Atmospheric pollutants are also included as their total column (TC) value, which become 2D variables with units kg m^{-2} and are here categorised as surface-level variables.

Variable	Units	Description
SURFACE-LEVEL METEOROLOGICAL VARIABLES		
2T	K	Temperature at 2 m above surface of land or sea
U10	m s^{-1}	Eastward component of wind at 10 m
V10	m s^{-1}	Southward component of wind at 10 m
WS	m s^{-1}	Wind speed at 10 m; equal to $(U10^2 + V10^2)^{1/2}$
MSL	Pa	Air pressure at mean sea level
ATMOSPHERIC METEOROLOGICAL VARIABLES		
U	m s^{-1}	Eastward component of wind
V	m s^{-1}	Southward component of wind
T	K	Temperature
Q	kg kg^{-1}	Specific humidity
Z	$\text{m}^2 \text{s}^{-2}$	Geopotential
SURFACE-LEVEL POLLUTION VARIABLES		
PM ₁	kg m^{-3}	Concentration of particulate matter with at most diameter 1 μm
PM _{2.5}	kg m^{-3}	Concentration of particulate matter with at most diameter 2.5 μm
PM ₁₀	kg m^{-3}	Concentration of particulate matter with at most diameter 10 μm
TC x	kg m^{-2}	Total column value of x
ATMOSPHERIC POLLUTION VARIABLES		
CO	kg m^{-3}	Concentration of carbon monoxide
NO	kg m^{-3}	Concentration of nitrogen monoxide
NO ₂	kg m^{-3}	Concentration of nitrogen dioxide
SO ₂	kg m^{-3}	Concentration of sulphur dioxide
O ₃	kg m^{-3}	Concentration of ozone
SURFACE-LEVEL WAVE VARIABLES		
10UN	m s^{-1}	Eastward component of natural wind at 10 m
10VN	m s^{-1}	Southward component of natural wind at 10 m
WSN	m s^{-1}	Natural wind speed at 10 m; equal to $(10UN^2 + 10VN^2)^{1/2}$
PP1D	s	Peak wave period
SWH	m	Significant wave height of total wave (wind waves plus swell)
MWP	s	Mean wave period of total wave
MWD	degrees	Mean wave direction of total wave (w.r.t. north)
SHWW	m	Significant height of wind waves
MPWW	s	Mean period of wind waves
MDWW	degrees	Mean direction of wind waves (w.r.t. north)
SHTS	m	Significant height of total swell
MPTS	s	Mean period of total swell
MDTS	degrees	Mean direction of total swell (w.r.t. north)
SWH1	m	Significant height of first swell partition
MWP1	s	Mean wave period of first swell partition
MWD1	degrees	Mean wave direction of first swell partition (w.r.t. north)
SWH2	m	Significant height of second swell partition
MWP2	s	Mean wave period of second swell partition
MWD2	degrees	Mean wave direction of second swell partition (w.r.t. north)

Table C3: Summary of the ten datasets used to pretrain Aurora in various configurations presented in this work. For every dataset, lists the resolution, timeframe that the dataset spans, the surface-level variables, the atmospheric variables, the number of pressure levels for the atmospheric variables, the number of time steps, and the total size.

Name	Resolution	Timeframe	Surf. variables	Atmos. variables	Levels	Steps	Size
ERA5	$0.25^\circ \times 0.25^\circ$	1979–2020	2T, 10U, 10V, MSL	U, V, T, Q, Z	13	368.18 k	105.50 TB
HRES-0.25 forecasts	$0.25^\circ \times 0.25^\circ$	2016–2020	2T, 10U, 10V, MSL	U, V, T, Q, Z	13	149.81 k	42.93 TB
IFS-ENS-0.25	$0.25^\circ \times 0.25^\circ$	2018–2020	2T, 10U, 10V, MSL	U, V, T, Q, Z	3	6.69 M	527.54 TB
IFS-ENS-0.25 mean	$0.25^\circ \times 0.25^\circ$	2018–2020	2T, 10U, 10V, MSL	U, V, T, Q, Z	3	133.71 k	10.55 TB
GFS forecasts	$0.25^\circ \times 0.25^\circ$	Feb 2015–2020	2T, 10U, 10V, MSL	U, V, T, Q, Z	13	354.40 k	101.56 TB
GFS T0	$0.25^\circ \times 0.25^\circ$	Feb 2015–2020	2T, 10U, 10V, MSL	U, V, T, Q, Z	13	8.64 k	2.48 TB
GEFS reforecasts	$0.25^\circ \times 0.25^\circ$	2000–2019	2T, MSL	U, V, T, Q, Z	7	2.96 M	454.61 TB
CMCC-CM2-VHR4	$0.25^\circ \times 0.25^\circ$	1950–2014	2T, 10U, 10V, MSL	U, V, T, Q	7	94.96 k	12.62 TB
ECMWF-IFS-HR	$0.45^\circ \times 0.45^\circ$	1950–2014	2T, 10U, 10V, MSL	U, V, T, Q, Z	7	94.96 k	4.75 TB
MERRA-2	$0.625^\circ \times 0.50^\circ$	1980–2020	2T, 10U, 10V, MSL	U, V, T, Q	13	119.81 k	5.58 TB
Total						10.97 M	1,268.12 TB

Table C4: Summary of the datasets used to fine-tune Aurora in various experiments presented in this work. For every dataset, lists the resolution, timeframe that the dataset spans, the surface-level variables, the atmospheric variables, the number of pressure levels for the atmospheric variables, the number of time steps, and the total size. Note that, in the wave experiment in Section 4, Aurora is fine-tuned on both wave and meteorological variables by lining up HRES-WAM-0.25 and HRES-0.25 T0.

Name	Resolution	Timeframe	Surf. variables	Atmos. variables	Levels	Steps	Size
CAMS reanalysis	$0.75^\circ \times 0.85^\circ$	2003–2021	2T, 10U, 10V, MSL, TC CO, TC NO, TC NO ₂ , TC SO ₂ , TC O ₃ , PM ₁ , PM _{2.5} , PM ₁₀	U, V, T, Q, Z, CO, NO, NO ₂ , SO ₂ , O ₃	13	55.52 k	3.65 TB
CAMS analysis	$0.40^\circ \times 0.40^\circ$	Oct 2017–May 2022	2T, 10U, 10V, MSL, TC CO, TC NO, TC NO ₂ , TC SO ₂ , TC O ₃ , PM ₁ , PM _{2.5} , PM ₁₀	U, V, T, Q, Z, CO, NO, NO ₂ , SO ₂ , O ₃	13	3.41 k	0.79 TB
HRES-WAM-0.25 an.	$0.25^\circ \times 0.25^\circ$	2016–2021	10UN, 10VN, WSN, PP1D, SWH, MWP, MWD, SHWW, MPWW, MDWW, SHTS, MPTS, MDTs, SWH1, MWP1, MWD1, SWH2, MWP2, MWD2	None	N/A	8.77 k	0.73 TB
HRES-0.25 T0	$0.25^\circ \times 0.25^\circ$	2016–2021	2T, 10U, 10V, MSL	U, V, T, Q, Z	13	8.77 k	2.51 TB
HRES-0.1 analysis	$0.10^\circ \times 0.10^\circ$	2016–2022	2T, 10U, 10V, MSL	U, V, T, Q, Z	13	10.23 k	18.30 TB
Total						86.69 k	25.98 TB

2. **Variable coverage:** Both datasets include our core set of meteorological variables (2T, U10, V10, MSL, U, V, T, Q) across 7 pressure levels (50, 250, 500, 600, 700, 850, 925 hPa), providing good vertical coverage of the atmosphere while maintaining consistency with our other data sources.
3. **Temporal extent:** These datasets span 1950-2014, offering exposure to long-term climate variability that complements our weather datasets. As shown in Supplementary G, this temporal diversity particularly benefits the prediction of extreme events, result in up to 35% error reduction in the tails of the distribution.

C.4 Training, validation, and test splits

For validation while pretraining, we use one year of IFS HRES at 0.25° resolution: 2020. Our test years are 2022 and 2023, depending on the dataset. Details of dataset splits can be found in Table C3, Table C4, and Table C5.

Motivation for CAMS analysis train–test split. CAMS undergoes frequent updates that dramatically affect the data distribution. Particularly notable are the updates in Sep 2017, when an issue was fixed that caused PM_{2.5} and PM₁₀ to be zero; July 2019, when the vertical resolution was increased; May 2021, which significantly alters the behaviour of the PMs; Dec 2022, which alters the background error covariances and significantly alters the distribution of the data; and June 2023, when CAMS is updated to the latest cycle C48R1 and is officially included as a chapter in the IFS documentation^[25]. Note that there were many more updates, but the mentioned ones were observed to be the most significant. Between these updates, the period May 2021 to Nov 2022 inclusive appears to be the most stable period. From this period, we use the last five months, Jun 2022 to Nov 2022 inclusive, for testing; and we include the first twelve months, Jun 2021 to May 2022 inclusive, for the fine-tuning data. We additionally include in the fine-tuning data all data going back to Oct 2017, after the issues with PM_{2.5} and PM₁₀ were fixed.

C.5 Dataset-specific postprocessing

For some datasets, like HRES-WAM, more careful postprocessing of the data is required. We detail all additional postprocessing steps here.

Postprocessing for HRES-WAM. In Section 4 we predict the neutral wind velocities. The neutral wind is related in a complex manner to the true wind velocity and represents the wind speed under neutral atmospheric conditions^[31]. It is critical in coupling wave and atmospheric models, and assimilation of measurements of wind speed over oceans, so we consider it an important variable for Aurora to predict after fine-tuning on this data.

HRES-WAM does not actually contain 10UN and 10VN, but instead the magnitude and the direction of the neutral wind speed, WIND and DWI respectively. We obtain 10UN and 10VN from WIND and DWI according to the following transformation:

$$10UN = -\sin(DWI) \cdot WIND, \quad 10VN = -\cos(DWI) \cdot WIND. \quad (C11)$$

Note the minus sign, which is necessary because the direction is where the wind *comes from*, not where it *goes towards*.

Absence of waves, e.g. above land or wherever there is sea ice, is represented with NaNs in the data. In addition, we sometimes find that significant wave heights and/or mean wave periods can be zero. Since zero significant wave height also means that there is no wave (or a particular wave component is absent), whenever significant wave height is zero, we set it and the corresponding mean wave period and mean wave direction to NaN. We do this separately for the total wave and every wave component, i.e. wind waves, total swell, swell one, and swell two. We verify that no spurious zeros remain after this processing step.

D Training methods

The overall training procedure is composed of three stages: (1) pretraining, (2) short-lead-time fine-tuning, (3) roll-out fine-tuning. We describe each of these stages in detail in the following subsections.

D.1 Training objective

Throughout pretraining and fine-tuning, we use the mean absolute error (MAE) as our training objective $\mathcal{L}(\hat{X}^t, X^t)$. Decomposing the predicted state $\hat{X}^t$ and ground truth state X^t into surface-level variables and atmospheric variables, $\hat{X}^t = (\hat{S}^t, \hat{A}^t)$ and $X^t = (S^t, A^t)$ (Supplementary A), the loss can be written as

$$\begin{aligned} \mathcal{L}(\hat{X}^t, X^t) = & \frac{\gamma}{V_S + V_A} \left[\alpha \left(\sum_{k=1}^{V_S} \frac{w_k^S}{H \times W} \sum_{i=1}^H \sum_{j=1}^W |\hat{S}_{k,i,j}^t - S_{k,i,j}^t| \right) \right. \\ & \left. + \beta \left(\sum_{k=1}^{V_A} \frac{1}{C \times H \times W} \sum_{c=1}^C w_{k,c}^A \sum_{i=1}^H \sum_{j=1}^W |\hat{A}_{k,c,i,j}^t - A_{k,c,i,j}^t| \right) \right], \end{aligned} \quad (\text{D12})$$

where w_k^S denotes the weight associated with surface-level variable k and $w_{k,c}^A$ denotes the weight associated with atmospheric variable k at pressure level c . The overall surface loss is weighted by $\alpha = \frac{1}{4}$, while the overall atmospheric loss is weighted by $\beta = 1$. Finally, the entire loss for a particular example is weighted by a dataset weight γ , which allows us to upweight the datasets with higher fidelity such as ERA5 and GFS-T0. Specifically, we use $\gamma_{\text{ERA5}} = 2.0$, $\gamma_{\text{GFS-T0}} = 1.5$, and set the rest of the dataset weights to 1. When training, we minimise the expected value of this loss computed over a mini-batch of samples.

Variable weighting for pretraining. During pretraining, we set $w_{\text{MSL}}^S = 1.5$, $w_{\text{U10}}^S = 0.77$, $w_{\text{V10}}^S = 0.66$, and $w_{\text{2T}}^S = 3.0$. For the atmospheric variables, for all pressure levels c , we set $w_{\text{Z},c}^A = 2.8$, $w_{\text{Q},c}^A = 0.78$, $w_{\text{T},c}^A = 1.7$, $w_{\text{U},c}^A = 0.87$, and $w_{\text{V},c}^A = 0.6$. The weights are chosen to balance the losses of the individual variables and have been inspired by the weights used by Bi et al. [2].

Variable weighting for concentrations of air pollutants. Air pollutants are extremely sparse. To balance these variables with the meteorological variables, we require a radically different approach. For the air pollution variables, we specify the weights such that the per-variable normalised MAE is roughly one.

For all air pollution variables, we compute the MAE for the persistence prediction of 12 h or 24 h, depending on whether the model predicts the difference with respect to the state 12 h ago or 24 h ago (Supplementary B.7). These persistence errors are computed on CAMS reanalysis data. We then set

$$w_v^S = \frac{\text{scale}_v}{\text{persistence MAE}_v}, \quad w_{v,c}^A = \frac{\text{scale}_{v,c}}{\text{persistence MAE}_{v,c}}. \quad (\text{D13})$$

Intuitively, multiplication by the scale first undoes the data normalisation and then dividing by the persistence MAE brings the normalised MAE to have a value of approximately one.

Variable weighting for wave variables. When fine-tuning Aurora for wave forecasting, we tune the weights to emphasise the more important and the more difficult variables. Specifically, we set $s_{\text{SWH}}^S = 2.0$, $s_{\text{MWP}}^S = 2.0$, $s_{\text{MWD}}^S = 2.0$, $s_{\text{PPID}}^S = 4.0$, $s_{\text{MPTS}}^S = 2.0$, and $s_{\text{MDTS}}^S = 2.0$.

Variable weighting for IFS T0 and analysis fine-tuning. During fine-tuning on IFS T0 0.25° and IFS analysis 0.1°, we slightly adjust the pretraining weights. For the surface-level variables, we set $w_{\text{MSL}}^S = 1.6$, $w_{\text{U10}}^S = 0.77$, $w_{\text{V10}}^S = 0.66$, and $w_{\text{2T}}^S = 3.5$. For the atmospheric variables, for all pressure levels c , we set $w_{\text{Z},c}^A = 3.5$, $w_{\text{Q},c}^A = 0.8$, $w_{\text{T},c}^A = 1.7$, $w_{\text{U},c}^A = 0.87$, and $w_{\text{V},c}^A = 0.6$.

Adjustments of the training loss for wave variables. For angle-valued variables, the MAE training loss is computed over the sine and cosine components of the angle (see Supplementary B.8). The sine and cosine components inherit the weighting of the original variables.

To deal with missing data, Aurora predicts the original variable along with a density channel (see Supplementary B.8). We compute the MAE over both the original variable and the density channel by also computing the density channel for the target variable following exactly the procedure outlined Supplementary B.8. The density channels inherit the weighting of the original variables. Note that the procedure sets all NaNs to zero, meaning that the MAE can be computed without risking the loss to become NaN.

D.2 Pretraining methods

All models are pretrained for 150k steps on 32 A100 GPUs, with a batch size of one per GPU. Our model sees about 4.8 million frames after 150k steps of pretraining, which takes approximately two and a half weeks. This roughly corresponds to 3 epochs over the C4 pretraining configuration that is outlined in Supplementary G. We use a (half) cosine decay with a linear warm-up from zero for 1k steps. The base learning rate is $5e-4$, which the schedule reduces by a factor $10\times$ at the end of training. The optimizer we use is AdamW^[51]. We set the weight decay of AdamW to $5e-6$. The only other form of regularisation we use is drop path (i.e., stochastic depth)^[52], with the drop probability set to 0.2. In order to make the model fit in memory, we use activation checkpointing for the backbone layers and we shard all the model gradients across GPUs. The model is trained using bf16 mixed precision.

Aurora pretraining in comparison. The training of Aurora is comparable to GraphCast (four weeks on 32 Cloud TPU v4 devices,^[3]) and faster than Pangu-Weather (about seven weeks using 64 GPUs,^[2]), and roughly comparable to AIFS (about one week with 64 GPUs in total^[71]). This demonstrates that Aurora achieves its improved performance and additional capabilities while maintaining training costs in line with other state-of-the-art models.

12-hour air pollution model. The 12-hour model used in the air pollution experiments was trained in exactly the same manner as described above, but for 80.5k steps instead of 150k steps.

D.3 Short lead-time fine-tuning

For each task we wish to adapt the pretrained Aurora model to, we start by fine-tuning the entire architecture through one or two roll-out steps (depending on the task and its memory constraints). In all cases, we use a task-dependent hyperparameter selection, which we describe below.

HRES 0.25° T0. We fine-tune the weights of the entire model for 8k training steps across 8 GPUs, with a batch size of 1 per GPU. At each iteration, we perform two roll-out steps and backpropagate through both of these steps. The model is optimised to minimise the MAE loss averaged across both roll-out steps. For this regime, we use a 1k step learning rate warm-up, followed by a constant learning rate of $5e-5$. We use the same weight decay as in pretraining and disable drop path. To ensure the model fits in memory for two roll-out steps, we also use activation checkpointing for the encoder and the decoder, along with gradient sharding as in pretraining.

HRES 0.1° analysis. We fine-tune the weights of the entire model for 12.5k steps across 8 GPUs, with a batch size of 1 per GPU. Due to the increased memory constraints at this higher resolution, we train the model only through a single step prediction. We use a 1k step learning rate warm-up, followed by a constant learning rate of $2e-4$. We set the weight decay to zero and disable drop path. To accommodate the higher memory requirements, we use activation checkpointing for all the layers of the model and use sharding for both the weights and the gradients.

CAMS 0.4° analysis. We train with single-step prediction, 12h in this case, and the batch size is fixed to 1 per GPU. We use a linear warmup of 100 steps from zero, but we do not use a learning rate schedule after that. We also use no weight decay, disable drop path, use activation checkpointing for all the layers of the model, and use sharding for the weights and gradients.

Fine-tuning on CAMS analysis data proceeds in two steps. In the first step, we fine-tune on CAMS reanalysis data using 16 GPUs for 22k steps at the high learning rate and then for 14.5k steps at the low learning rate. The high learning rate is $1e-3$ for the encoder patch embeddings of *only* the new pollution variables and $1e-4$ for the rest of the the network. The low learning rate is $1e-4$ for the encoder patch embeddings of only the new pollution variables and $1e-4$ for the rest of the network. To ensure maximum transfer from the CAMS reanalysis data to the CAMS analysis data, the CAMS reanalysis data is regridded to the resolution of CAMS analysis data, 0.4° . In the second step, we fine-tune on CAMS analysis data using 8 GPUs for 7.5k steps at the high learning rate and finally for 5.5k steps at the low learning rate. The final model is fine-tuned for 49.5k steps in total.

HRES-WAM 0.25° analysis. We fine-tune the weights of the entire model for 14k steps across 8 GPUs, with a batch size of 1 per GPU. For the learning rate, we use a linear warmup of 500 steps to $1e-3$ for the encoder and decoder parameters of the new wave variables and $3e-4$ for the rest of the network. The

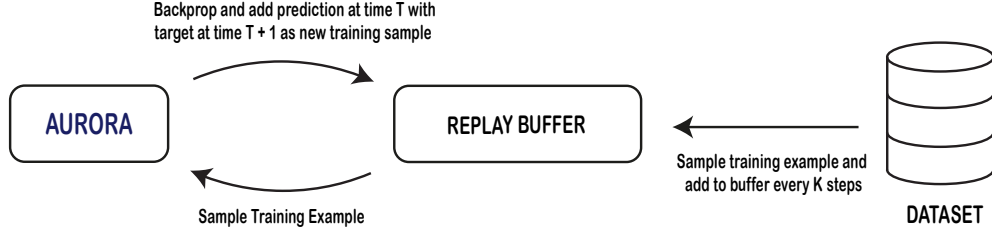


Fig. D2: Diagram of the roll-out fine-tuning procedure of Aurora. The replay buffer is initially populated with samples from the dataset. At each fine-tuning step, the model fetches a training sample from the replay buffer, performs a training step, and then it adds this new prediction (together with its next step target from the dataset) to the replay buffer. Every K steps, the replay buffer is refreshed with a new training sample from the dataset. Since the replay buffer is generally much smaller than the dataset size, this ensures that enough samples from the dataset are seen. Overall, this procedure allows the model to train fast on an evolving distribution of autoregressive roll-out steps coming from a mixture of model versions. This avoids the need to run expensive roll-out procedures at each step.

learning rate is then annealed to $1e-5$ according to a cosine schedule for 30 k steps (the run was stopped at 14 k steps). Afterwards, the training data was restricted to the range Jul 2018–2021: on 5 June 2018, IFS cycle 45r1 was implemented, which significantly changed the distribution of the data by coupling of the three-dimensional ocean and atmosphere. For this adjusted training data range, we further fine-tune the weights of the entire model for 10 k steps across 8 GPUs, with a batch size of 1 per GPU. For the learning rate, we again use a linear warmup of 500 steps to $5e-5$ for all parameters. The learning rate is then annealed to $1e-5$ according to a cosine schedule for 10 k steps. The final model is fine-tuned for 24 k steps in total.

D.4 Roll-out fine-tuning

To ensure long-term multi-step dynamics, AI models typically fine-tune the model specifically for roll-outs. Backpropagating through the autoregressive roll-outs for a large number of steps is unfeasible for a 1.3B parameter model such as Aurora. This is particularly true at 0.1° resolution, where even a single step roll-out requires close to the memory limit of an A100 GPU with 80GB of memory.

We use Low Rank Adaption (LoRA) [53] layers for roll-out fine-tuning all the linear layers involved in the self-attention operations of the backbone. This allows us to take advantage of the large model size and the fact that it can be easily adapted once pretrained. That is, for each linear transformation W involved in the Swin self-attention layers, we learn low-rank matrices A, B to modulate the outputs of W for an input x via $Wx + BAx$. For more details, see Hu et al. [53].

Furthermore, to avoid any memory increases compared to single-step fine-tuning, we use the “pushforward trick” introduced in Brandstetter et al. [54], where gradients are propagated only through the last roll-out step. We run this at scale by using an in-memory replay buffer to avoid delays with generating long roll-outs on each training step, similarly to how it is used in deep reinforcement learning [55, 56] (Figure D2). At each training step, the model samples an initial condition from the replay buffer, computes a prediction for the next time step, then adds this new prediction back to the replay buffer. We periodically fetch fresh initial conditions from the dataset and add them to the replay buffer (i.e., the dataset sampling period). This procedure allows the model to train at all roll-out steps without extra memory or speed penalties.

HRES 0.25° analysis. We use 20 GPUs to fine-tune the LoRA layers for 13 k steps, each with a buffer size of 200. This results in a total replay buffer size of 4000 samples. We use a dataset sampling period of 10 steps. To ensure the model learns to predict the early steps well (i.e., shorter lead-times) before attempting to predict the later time steps, we use a schedule where for the first 5 k steps, we only keep predictions up to 4 days ahead in the buffer. The 4–10 day lead times are allowed in the buffer only after 5 k steps. We use a constant learning rate of $5e-5$.

HRES 0.1° analysis. Since the 0.1° data is $6.25\times$ larger than 0.25° data, we use 32 GPUs with a buffer size of 20 on each GPU. This is the maximum we can fit in the CPU memory of each node (i.e., 880 GB). We use a dataset sampling period of 10 steps. We train the LoRA weights of the model for 6.25 k steps using a constant learning rate of $5e-5$.

CAMS 0.4° analysis. We use 16 GPUs with a buffer size of 200 on each GPU and a dataset sampling period of 10 steps. We train the LoRA weights of the model for 6.5k steps using a constant learning rate of $5e-5$.

HRES-WAM 0.25° analysis. We use 8 GPUs with a buffer size of 100 on each GPU and a dataset sampling period of 10 steps. We train the LoRA weights of the model for 6k steps using a constant learning rate of $5e-5$.

E Data infrastructure

Data loading represented one of the major technical challenges when training Aurora. First, due to the sheer size of one datapoint (close to 2GB for 0.1° data), loading the data efficiently on the GPUs becomes extremely challenging. Second, training Aurora on many heterogeneous datasets with different resolutions, variables, and pressure levels, makes the task even more difficult as batching together samples from different datasets becomes nontrivial and the workload of the GPUs must be balanced. In what follows, we describe the data storage and data loading infrastructure we built to handle these two problems.

Data storage and preprocessing. Because of the large size of the datasets, the data cannot be stored locally, so must be stored with a cloud solution. We use Azure’s blob storage. To ensure that data can be efficiently downloaded from blob storage, we perform several optimizations in how the data is stored:

1. We co-locate data and compute to minimise latency and costs.
2. We preprocess each dataset into small chunks/files in order to make sure that a worker does not have to download substantially more data than necessary to access a given sample from the dataset. This is important because the raw form of many datasets stores a month (or more) of data into a single file. We found the Zarr format to particularly useful for this.
3. For large datasets (like ERA5), we compress the files in order to minimise network bandwidth when downloading samples.
4. For large datasets, we also make sure that all the variables for a given timestep are stored in the same file. This minimises the number of concurrent file downloads that have to be performed to fetch one sample.

Data loading. To load heterogeneous data efficiently, we developed an advanced multi-source data loading pipeline that can satisfy these requirements (Figure E3). Each dataset is instantiated by a `yaml` config specifying how the data from that dataset should be loaded (e.g., dataset location, which lead time should be used, the number of past steps to be supplied as input, etc.). Then, each dataset generates a stream of `BatchGenerator` objects. These are lightweight objects containing the necessary metadata (e.g., file paths) to download and generate at least one batch from the associated dataset. The streams of `BatchGenerator` objects from all the datasets are combined into one stream via sampling, which allows us to adjust the relative proportions of the datasets within each epoch. This is followed by shuffling all the `BatchGenerator` objects and sharding them across GPUs. Up to sharding, all these lightweight operations are run identically on all the GPUs using the same random seeds.

The expensive operations of the data loading pipeline follow the sharding. The `BatchGenerator` objects sent to each GPU are unpacked into actual batches of data that can be used for training and inference. This unpacking process, consists of a series of sequential transformations that are specific to each dataset and the particular way it instantiates its `BatchGenerator` objects. In general, the sequence of transformation includes: downloading some (compressed) dataset files, decompressing them, reading the data from the files, generating the samples, mapping the variable names to a canonical set of names, performing data checks (e.g., handling NaNs), batching, and, finally, loading batches to the device. All of these operations are performed across multiple workers.

This data loading pipeline enables efficient training on multiple heterogeneous datasets. Performing batched computations is straightforward since only samples from the same dataset are batched together, and therefore there are no any additional constraints imposed on the model. To balance workloads across GPUs and workers, one can use bigger batch sizes for the smaller datasets and smaller batch sizes for the bigger datasets, since samples from different datasets have different sizes.

Dehghani et al.^[72] approach the problem of training ViTs on multi-resolution images at the model layer level, by concatenating tokens from different images and using masking to prevent tokens from different

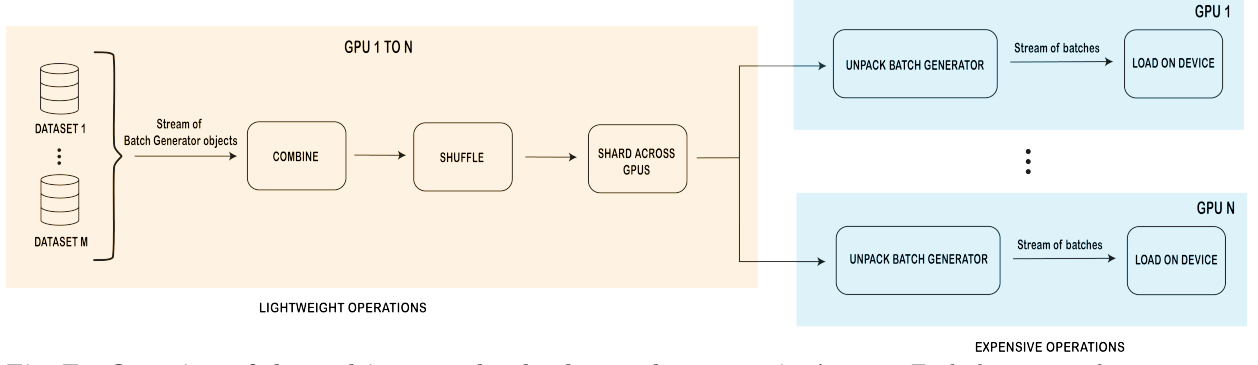


Fig. E3: Overview of the multi-source dataloader used to pretrain Aurora. Each dataset produces a stream of lightweight **BatchGenerator** objects. The streams are combined, shuffled, and sharded across GPUs. After sharding, each GPU receives a fraction of the **BatchGenerator** objects, which are then unpacked. This step performs the expensive procedures like downloading the data associated with the object from Azure blob storage. This procedure ensures that only samples from the same together are batched together and provides the flexibility to perform custom per-dataset operations.

images from communicating during self-attention operations. However, their approach imposes a constraint on the model architecture and it is difficult to implement in more sophisticated ViT architectures such as the Swin Transformer^{[19] [50]}. Masked self-attention, even if implemented with sparse matrix multiplications, is not able to achieve the expected theoretical performance due to the lack of semi-structured sparsity support on current hardware. Therefore, another advantage of our data loading pipeline is that it allows for the freedom to experiment on the model architecture side.

F Verification metrics

F.1 Main evaluation metrics

The main metrics we use to measure the performance of Aurora against other methods are the root mean square error (RMSE) and the anomaly correlation coefficient (ACC). These metrics use a latitude weighting because the variables and targets live on a non-uniform grid,

$$w(i) = \frac{\cos(\text{lat}(i))}{\frac{1}{H} \sum_{i'=1}^H \cos(\text{lat}(i'))}, \quad (\text{F14})$$

which downweights smaller cells (closer to the poles) and upweights bigger cells (closer to the equator). The weights are normalised to unit mean. While various versions of this weighting are used in practice, the form used here is the one from Rasp et al.^[73].

Root mean square error (RMSE). The latitude-weighted RMSE measures the magnitude of the errors between the predictions and the ground truth,

$$\text{RMSE} = \frac{1}{T} \sum_{t=1}^T \sqrt{\frac{1}{H \times W} \sum_{i=1}^H \sum_{j=1}^W w(i) (\hat{X}_{i,j}^t - X_{i,j}^t)^2}, \quad (\text{F15})$$

where t indexes over the sample datasets, i, j index over the latitude and longitude of each image, and $w(i)$ is the latitude weighting factor.

Adjustments of the RMSE for wave variables. For angle-valued variables, when computing the difference between two angles, we need to account the fact that 0° is equal to 360° , meaning that e.g. the absolute difference between 10° and 350° should be 20° and not 340° . We account for this by replacing $|d| = |\hat{X}_{i,j}^t - X_{i,j}^t|$ with $\min(|d|, 360^\circ - |d|)$.

In addition, when computing the RMSE, because wave variables have missing data and Aurora can also predict the absence of data (see Supplementary B.8), either or both $\hat{X}_{i,j}^t$ and $X_{i,j}^t$ may be undefined. In

Equation (F15), we compute the average only over points where the target $X_{i,j}^t$ is defined. When doing this, we correctly renormalise the latitude weights to average to one. Wherever the target $X_{i,j}^t$ is defined but the prediction $\hat{X}_{i,j}^t$ is undefined, we replace $\hat{X}_{i,j}^t$ with the normalisation mean.

Anomaly correlation coefficient (ACC). The ACC measures the correlation between the deviation of the prediction and the ground truth from the daily climatology (i.e., the daily mean of a variable for that day of the year). It takes the form

$$\text{ACC} = \frac{1}{T} \sum_{t=1}^T \frac{\sum_{i=1}^H \sum_{j=1}^W w(i) (\hat{X}_{i,j}^t - C_{i,j}^t) (X_{i,j}^t - C_{i,j}^t)}{\sqrt{\left[\sum_{i=1}^H \sum_{j=1}^W w(i) (\hat{X}_{i,j}^t - C_{i,j}^t)^2 \right] \left[\sum_{i=1}^H \sum_{j=1}^W w(i) (X_{i,j}^t - C_{i,j}^t)^2 \right]}}, \quad (\text{F16})$$

where $C_{i,j}^t$ is the daily climatology for location i, j for the day of the year corresponding to time t . Our daily climatology is computed based on the climatology provided by Rasp et al. [60].

F.2 Extreme weather metrics

Thresholded RMSE. The thresholded RMSE measures the magnitude of the errors between the predictions and the ground truth. We apply an additional threshold here to indicate which latitude-longitude gridpoints should be included in the sum. This can be written as:

$$\text{RMSE}_g = \frac{1}{T} \sum_{t=1}^T \sqrt{\sum_{i=1}^H \sum_{j=1}^W \tilde{w}_{i,j}^{g,t} (\hat{X}_{i,j}^t - X_{i,j}^t)^2}. \quad (\text{F17})$$

When computing RMSEs above a threshold, the weighting factor for the right-sided thresholds (plotted on the right side of the relevant plots) is then defined by:

$$\tilde{w}_{i,j}^{g,t} = \frac{\mathbb{1}(X_{i,j}^t > b_{i,j}^g) w(i)}{\sum_{i=1}^H \sum_{j=1}^W \mathbb{1}(X_{i,j}^g > b_{i,j}^g) w(i)}. \quad (\text{F18})$$

For the left-sided thresholds, this is instead defined by:

$$\tilde{w}_{i,j}^{g,t} = \frac{\mathbb{1}(X_{i,j}^t < b_{i,j}^g) w(i)}{\sum_{i=1}^H \sum_{j=1}^W \mathbb{1}(X_{i,j}^g < b_{i,j}^g) w(i)}. \quad (\text{F19})$$

This incorporates the threshold b defined as:

$$b_{i,j} = \mu_{i,j} + g \cdot \sigma_{i,j}, \quad (\text{F20})$$

where μ and σ are the mean and standard deviation of the ERA5 reanalysis data over all training years, computed for each latitude-longitude point i, j . Here, g is a factor that is varied linearly for both positive and negative values and is used to obtain the thresholded RMSEs at a number of points, and in the figures we plot the resulting curves. The latitude weighting factor $w(i)$ is as defined in Equation (F14). The value is computed for every example frame, and this set of values, sampled with replacement, is used to bootstrap 95% confidence intervals by recomputing Equation (F17). For $g > 0$, we plot RMSE_g computed using Equation (F18), and for $g < 0$, we plot RMSE_g computed using Equation (F19).

The plots show a discontinuity at zero; this is to improve their readability. We remove the lower half of the right-hand side of the curve and the upper half of the left-hand side of the curve. However, it is important to point out that RMSE_g is directly comparable to RMSE: RMSE_g computed using Equation (F19) follows $\text{RMSE}_g \rightarrow \text{RMSE}$ as $g \rightarrow \infty$, while RMSE_g computed using Equation (F18) follows $\text{RMSE}_g \rightarrow \text{RMSE}$ as $g \rightarrow -\infty$.

G Data diversity and model scaling improve atmospheric forecasting

A key tenet of the foundation model paradigm is that performance improves in a significant and predictable fashion as the data and the model are scaled up^[74,81]. We provide evidence that these findings from natural language processing and computer vision also hold for atmospheric forecasting.

Most medium-range AIWP approaches have employed a single dataset in their training strategy, usually ERA5 at 0.25° resolution. Training Aurora on multiple datasets allows us to examine the benefits of including diverse datasets in pretraining. To that end, we pretrain the main Aurora model on four different dataset configurations (labelled C1–C4) and compare (1) the RMSE of the models across different variables and levels on ERA5 2021 (Extended Data Figure 1a) and (2) the RMSE of the models for extreme values at the surface on HRES 2022 (Extended Data Figure 1b). Neither of these two years of data were included in pretraining and no further fine-tuning was applied to these models.

Adding climate simulation data consisting of CMIP6-CMCC and CMIP6-IFS-HR alongside high-quality ERA5 reanalysis data (C2) results in lower RMSEs than the ERA5-only model (C1) across all the groups of variables (except for 2T) (Extended Data Figure 1a). Although these data are of lower fidelity compared to reanalysis datasets like ERA5, including climate simulations increases the diversity of the data and, as our results suggest, generalization performance. When also including IFS ensemble data, IFS HRES forecast data, and the IFS ensemble mean (C3), the performance improves even further at the surface and around the levels present in the IFS-ENS data, which are highlighted in the figure. Since the improvements seen with C3 diminish in other parts of the atmosphere, we also test an alternative configuration (C4), which provides better coverage of the entire atmosphere. Additionally, it limits the samples used in training for the forecasts datasets to one-day lead-times to increase the data quality. This last configuration performs best overall and, for this reason, it is also the configuration used for the main results reported in this paper. Crucially, notice that the downstream validation dataset we use here is a test year of ERA5 itself, so we see positive transfer when including additional datasets even in the case where the downstream tasks overlaps perfectly with ERA5.

Including additional data in pretraining not only improves the RMSEs in aggregate, but it also leads to significant improvements (up to 35%) in the tails of the distribution at the surface on HRES 2022. Extended Data Figure 1b shows RMSEs computed across all cases where the underlying variable is higher than a threshold value (see Supplementary F.2) for five surface-level variables. As before, C2 improves over C1, while C3 and C4 further improve upon C2. In all instances, the improvements compared to the ERA5-pretrained model (C1) are mostly in the extremes. We postulate that this is because climate simulations and diverse data sources expose the model to more rare events than historical data can provide, which ultimately improves the model robustness in these regimes.

Overall, these results provide evidence that using a heterogeneous group of weather and climate datasets, of mixed fidelity, can improve forecasting performance. In Supplementary G.1 we verify that these findings also hold across the individual atmospheric variables and include additional results on HRES 2022. Finally, we discuss one other dataset configuration that does not yield performance improvements.

Orthogonal to the data scaling dimension, we test the impact of model scaling on four pretrained Aurora configurations, each corresponding to a different model size: 113 M, 290 M, 660 M and 1.3 B parameters (see Supplementary B.9). All models are pretrained on the dataset mixture from C4 and continuously validated on IFS analysis data. After the initial training phase, bigger models achieve lower validation losses for the same number of GPU hours (Extended Data Figure 1c). The bigger models achieve this whilst being trained on fewer data samples (measured in number of tokens in the figure). When fitting a power law to the performance of the three models at the 5k GPU hours mark, we obtain a scaling law suggesting that validation performance improves by approximately 6% for every 10× increase in model size. Note that the validation performance on IFS analysis data is computed without fine-tuning on IFS analysis data, to limit computational costs.

G.1 Effects of data diversity

The following two subsections offer an overview of the main reasons why pretraining on multiple datasets can have beneficial effects and a more detailed discussion of our results together with additional experiments.

G.1.1 Why increase the data diversity?

The use of several different dataset to train Aurora can help along several dimensions:

1. Climate simulations expose Aurora to entirely new simulated weather patterns not present in observational datasets, effectively increasing the variety and size of our training data. These simulations capture potential future atmospheric states and extreme events that may be underrepresented in historical data.
2. Forecast datasets provide additional examples of realistic weather patterns that did not necessarily occur, again increasing the diversity of training examples. These forecasts help the model learn about plausible atmospheric evolution paths beyond what is captured in analysis data alone.
3. While GFS differs from IFS in its implementation, it models similar underlying physics with different parameterizations. This variation helps Aurora learn more robust representations of atmospheric dynamics that can avoid inheriting the biases of an specific NWP model.
4. Different datasets have varying spatial resolutions, numbers of pressure levels, and quality of physics models. By training on this heterogeneous mix, Aurora learns to extract the best features from each source, e.g., high-resolution details from some datasets, better vertical structure from others, and improved physical consistency from the most sophisticated models.

Overall, pretraining on the output of numerical simulators can be seen as an analogous practice to the training of large language models on synthetically generated data^[75], a standard practice in the field.

G.1.2 Detailed analysis

Here we examine the effects of increasing data diversity across all variables and levels in more detail. Besides the four dataset configurations C1–C4 discussed before, we also consider one additional dataset configuration based on all the datasets described in Table C3, which we label as C5. This latter configuration contains all the datasets included in C3 and additionally Merra-2, GFS-T0 analysis, GFS forecasts and GEFS reforecasts. We evaluate all pretrained jobs both on ERA5 2021 (Figure G5) and HRES-T0 analysis 2022 (Figure G4). No further fine-tuning was performed after the pretraining.

On ERA5, we observe that the trend described before holds across all the individual atmospheric variables. C2 improves over C1 almost universally, C3 improves further over C2 on most atmospheric variables for those levels that are covered by the IFS-ENS and IFS-ENS-Mean datasets (which are the biggest datasets in C3). Similarly, C4 improves over C2, as expected due to the inclusion relationship between them. Furthermore, C4 also generally performs better than C3.

Perhaps surprisingly, we notice that C5, which is a superset of all the other configurations, does not perform as well as other configurations and often does worse than the ERA5 job. There are couple of potential hypothesis behind this. A first possibility is that some of the newly introduced datasets in C5 do not align well with the evaluation dataset (ERA5). For instance, Merra-2 is the lowest resolution dataset in the entire pretraining mix and its addition might harm performance on a high resolution dataset like ERA5. At the same time, the GEFS reforecast dataset only includes three levels in the lower part of the atmosphere (850, 925, 1000 hPa), so it does not provide a good coverage of the atmosphere. Another possibility is that including such a large amount of data likely requires significantly more training than 150 k steps to yield good performance.

On HRES T0 2022, the trend for the surface variables looks similar to ERA5, with the exception of 2T. Since HRES T0 is missing an extra assimilation step for the surface, 2T in HRES T0 exhibits different biases compared to 2T in ERA5. We believe including any kind of IFS forecast data in pretraining helps alleviate this distribution mismatch, leading to substantial improvements. In terms of the atmosphere, performance also generally improves across most variables with the addition of more data, but the order of the configurations fluctuates more. One interesting case is specific humidity (Q), where for unknown reasons, C3 and C4 perform significantly worse than the ERA5-pretrained model on some levels. This could be a

potential explanation for why Aurora performs slightly worse than GraphCast at certain lead times and levels of Q in the scorecard from Figure H6.

Finally, we note that on both test datasets, the biggest improvements from additional data on the atmospheric variables are obtained for the geopotential (Z). This is likely explained by the fact that the physical equations describing the evolution of the geopotential are well-understood and, therefore, the NWP-simulations we add in pretraining provide very high-quality data for this variable. Interestingly, the geopotential also improves universally on C2, despite being absent from the new datasets in this configuration. This demonstrates the ability of the model to learn useful patterns even from incomplete variable sets.

G.2 Model scaling laws for surface-level variables

In Extended Data Figure 1c, we saw that bigger models obtain lower validation loss for the same number of GPU hours. In Extended Data Figure 2, we show that the same conclusion is also true for every surface-level variable individually. Because the training loss is an aggregate of approximately 70 different variables (four surface-level variables and five atmospheric variables at thirteen different pressure levels), how individual variables scale as model size increases is more noisy than the aggregate loss. Nevertheless, we observe clear scaling behaviour for every surface-level variable individually.



Fig. G4: Effects of data scaling measured with performance on HRES T0 2022 across five dataset configurations. The percentages are relative to a model pretrained on only ERA5. **a:** The RMSE for every surface-level variable. **b:** The RMSE for every pressure level variable aggregated across all atmospheric variables. The greyed out range indicates the pressure levels present in the IFS ensemble. **c:** For every atmospheric variable, the RMSE for every pressure level.

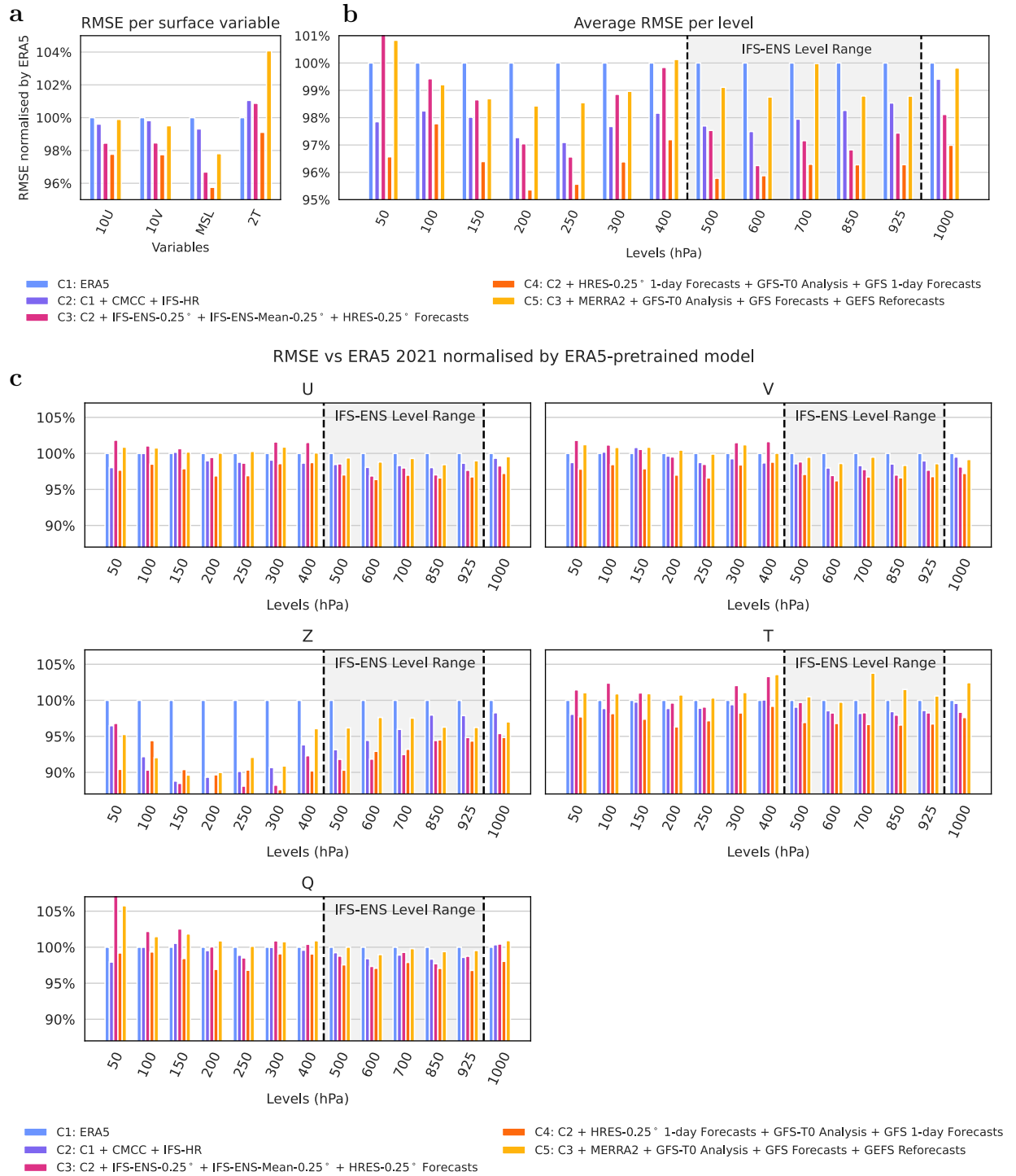


Fig. G5: Effects of data scaling measured with performance on ERA5 2021 across five dataset configurations. The percentages are relative to a model pretrained on only ERA5. **a:** The RMSE for every surface-level variable. **b:** The RMSE for every pressure level variable aggregated across all atmospheric variables. The greyed out range indicates the pressure levels present in the IFS ensemble. **c:** For every atmospheric variable, the RMSE for every pressure level.

H Comparison against GraphCast, Pangu, and IFS HRES at 0.25° resolution

The vast majority of AI models for weather prediction have been (pre)trained at 0.25° resolution on a single dataset. As an additional validation of the benefits of fine-tuning a large model pretrained on many datasets, we compare Aurora against operational GraphCast^[3], which is pretrained only on ERA5 and fine-tuned to IFS-HRES analysis, rendering it the closest setup to Aurora. Moreover, operational GraphCast is currently considered the state-of-the-art in operational AI weather forecasting at 0.25° and lead times up to 5 days^[60]. As an additional datapoint, we include in the comparison Pangu weather^[2], another AI model which is widely considered competitive. For consistency we apply the same evaluation procedure to all three AIWP models. We also include IFS HRES forecasts in this comparison, which is the gold standard in operational numerical weather prediction. We show that Aurora outperforms both when measured against IFS analysis, weather station observations, and extreme values.

H.1 Experimental setup

This comparison uses the HRES-T0 2022 dataset as a test year and is carried out at the standard 0.25° resolution used in AIWP. To this end, Aurora is fine-tuned on HRES-T0 “analysis” data for years 2016-2021 (see Supplementary C.2), the same years that operational GraphCast was fine-tuned on. In order to ensure a consistent comparison against operational NWP systems, both GraphCast and Aurora are evaluated using daily 00 and 12 UTC initialisation times from the HRES-T0 dataset provided by WeatherBench2. As recommended, we treat HRES-T0 as the ground truth targets for this evaluation. To facilitate a complete comparison between models across different variables, pressure levels and lead times, we compute and visualize RMSE skill scores in the form of *relative RMSE difference between the models that are compared*, normalized by the RMSE score of the model we compare against. We also include comparisons based on the ACC metric in Supplementary I.

Comment on the evaluation dataset. It is important to clarify that this evaluation procedure does not compare against ERA5 (as seen in GraphCast) for two primary reasons:

1. This procedure evaluates the system under operational conditions following the protocol established by ECMWF^[45]. This represents a real-world operational evaluation. Note that GraphCast^[3] had to use a rather *ad hoc* procedure to attempt to compensate for the fact that ERA5 was used on the input and output side for their model (see section 5.2.1 of the GraphCast supplementary material).
2. The IFS HRES analysis is closer to the ground truth state of the weather than ERA5. This is because ERA5 uses an NWP model that was operational in 2017^[45]. In the seven years since, IFS has undergone a large number of improvements.

We do not use the original WeatherBench2 evaluation protocol because until recently this used comparisons at 1.4° of ERA5 as inputs and targets. Not only is this a low spatial resolution, ERA5 inputs are reanalyses and as such do not reflect available information at the time of a forecast because they incorporate observational data from the future. Moreover, as mentioned above, ERA5 is considered less accurate to the ground truth than IFS HRES analysis. More recently, WeatherBench2 started to support the evaluation protocol outlined above.

H.2 Aurora versus GraphCast at 0.25°

Aurora consistently outperforms GraphCast^[3] across the vast majority of target variables, pressure levels, and lead times (Figure H6). The performance gains are most pronounced at lead times past 3 days, as well as for the upper atmospheric levels where we see a reduction in RMSE up to 40%. Large improvements up to 10-15% are observed at both short and long lead times, especially for variables such as temperature (T), geopotential height (Z) and wind velocity components (U, V). The smallest gains are observed for specific humidity, where GraphCast tends to yield more accurate predictions at shorter lead times (1–3 days) for the surface and lower atmospheric levels.

At longer lead times, Aurora’s forecasts more closely resemble the ensemble mean than GraphCast (see Supplementary I.8), but Aurora also outperforms GraphCast at short lead times on U, V, T, and Z. Taken together, we postulate that these results are a collective consequence of Aurora’s scale, both in terms of

architecture design and training data corpus, as well as its pretraining and fine-tuning protocols. We discuss the origin of Aurora’s performance in more detail below.

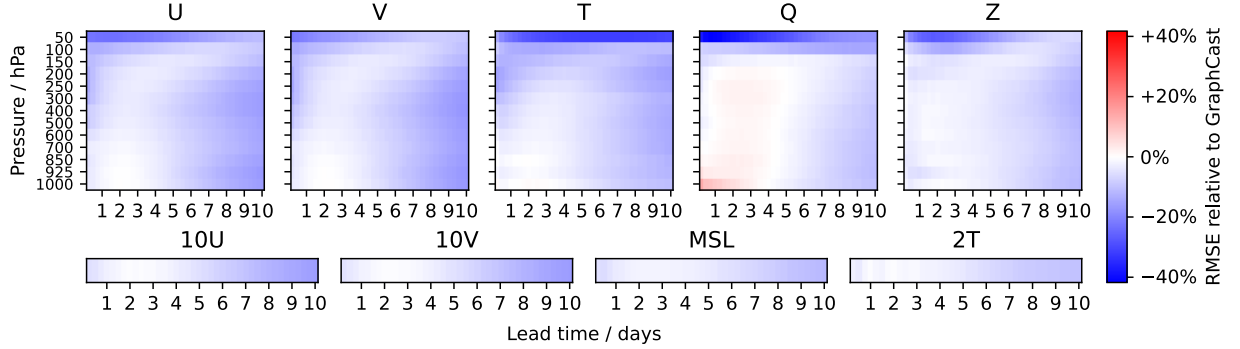


Fig. H6: Scorecard comparing Aurora and GraphCast. Aurora matches or outperforms GraphCast at 0.25° on 94% of targets. Aurora has the largest gains (40%) over GraphCast in the upper atmosphere, where GraphCast performance has notable room for improvement. The two models are closest to each other in the lower atmosphere at the 2–3 day lead time, which corresponds to the lead time GraphCast was roll-out fine-tuned on.

For a more detailed comparison against GraphCast, we include various additional roll-out plots: RMSEs for the headline metrics (Figure H7), RMSEs in the lower atmosphere (Figure H8), ACCs in the lower atmosphere (Figure H10), RMSEs in the upper atmosphere (Figure H9), and ACCs in the upper atmosphere (Figure H11).

We believe Aurora’s improved performance over GraphCast can be attributed to several factors:

1. **Data diversity:** Our ablation studies in Supplementary G demonstrate clear benefits from diverse training data, with Extended Data Figure 1a showing improved accuracy across variables when including climate simulations and multiple weather datasets. Extended Data Figure 1b provides concrete evidence that this diversity leads to up to 35% error reduction in extreme value prediction compared to training on ERA5 alone.
2. **Model size and architecture:** Aurora’s larger size (1.3B vs GraphCast’s 37M parameters) translates to measurable performance gains, as shown by our scaling experiments in Extended Data Figure 1c where validation performance improves systematically with model size. Our flexible encoder architecture (detailed in Supplementary B.1) enables efficient processing of heterogeneous inputs, which is crucial for learning from diverse datasets.
3. **Foundation model approach:** The effectiveness of our pretraining strategy is demonstrated in Supplementary Supplementary G.1, where Figure G4 and Figure G5 show consistent improvements across variables and pressure levels as we include more diverse datasets in pretraining. This validates our approach of leveraging knowledge from multiple domains to improve overall performance.
4. **Advanced fine-tuning strategy:** Our two-stage fine-tuning process, including the use of LoRA for roll-out fine-tuning (Supplementary Supplementary D.4), provides better control of the model’s behavior at different lead times. The benefits of this approach are evident in our comparison against GraphCast (Figure H6), where Aurora shows improved performance across the vast majority of targets, particularly at longer lead times.

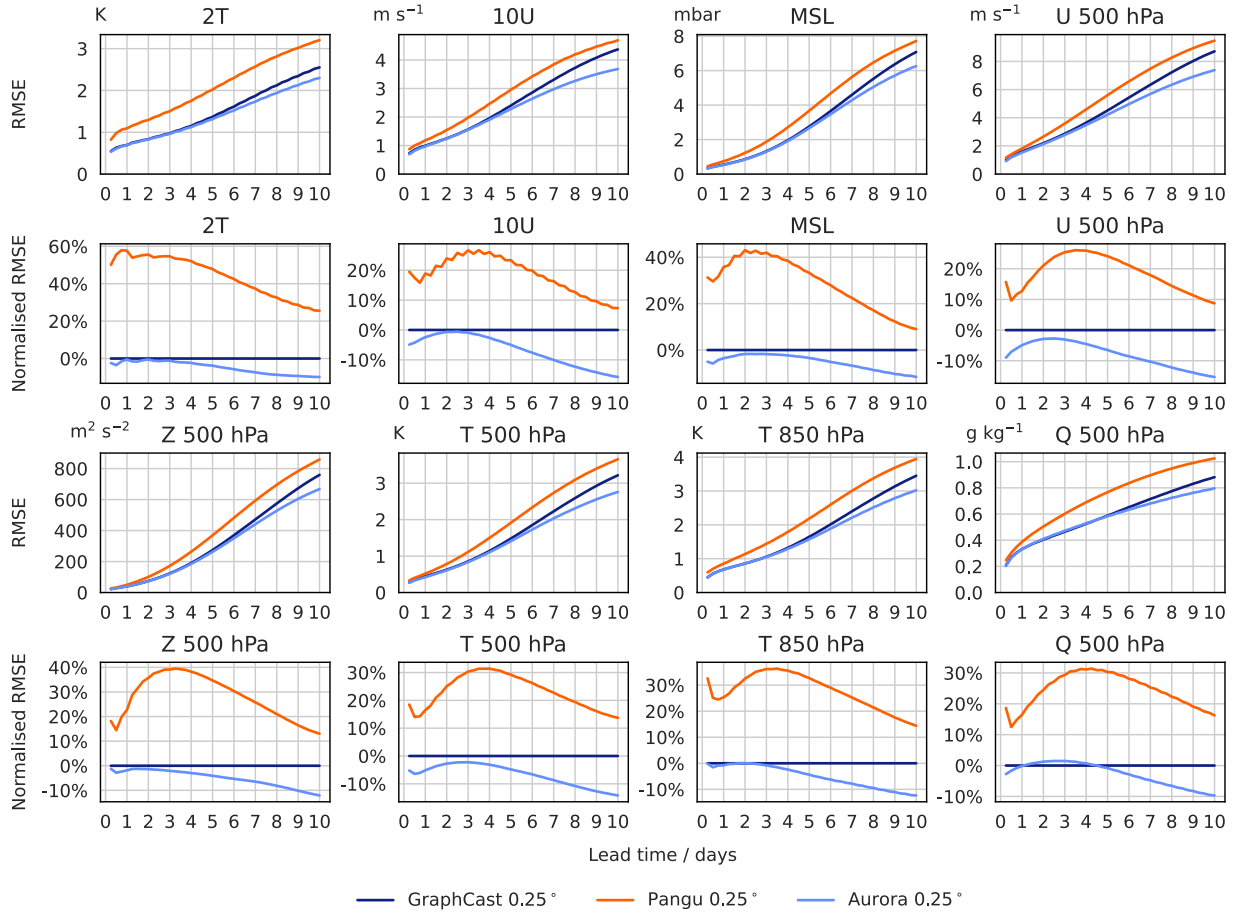


Fig. H7: Root mean square error (RMSE) of Aurora compared to GraphCast for the headline variables. Shows both unnormalised and normalised RMSEs at 0.25°, where GraphCast is used as the baseline. The Pangu weather results at 0.25° are included for reference.

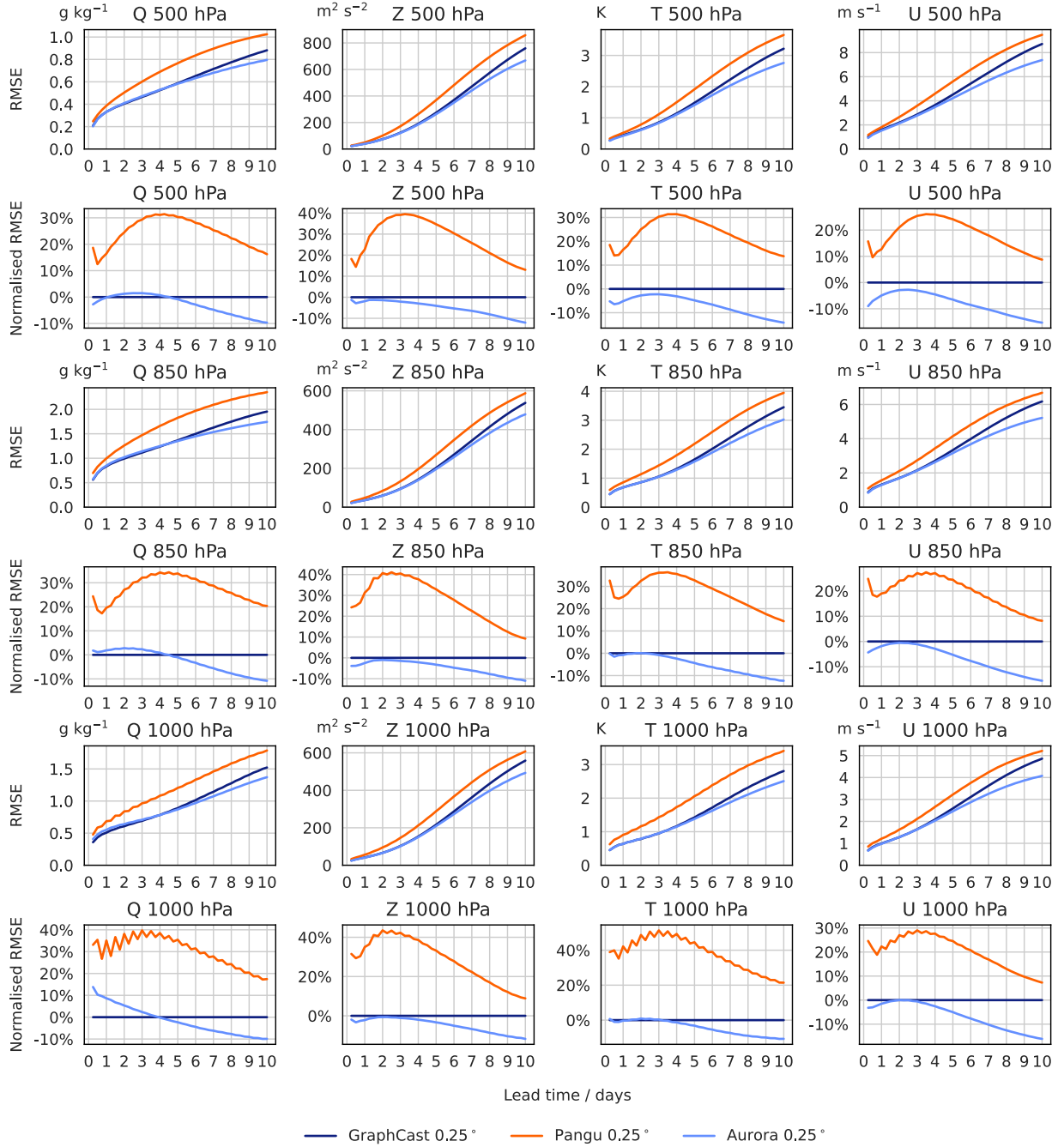


Fig. H8: Root mean square error (RMSE) of Aurora compared to GraphCast in the lower atmosphere. Shows both unnormalised and normalised RMSEs at 0.25°, where GraphCast is used as the baseline. The Pangu weather results at 0.25° are included for reference.

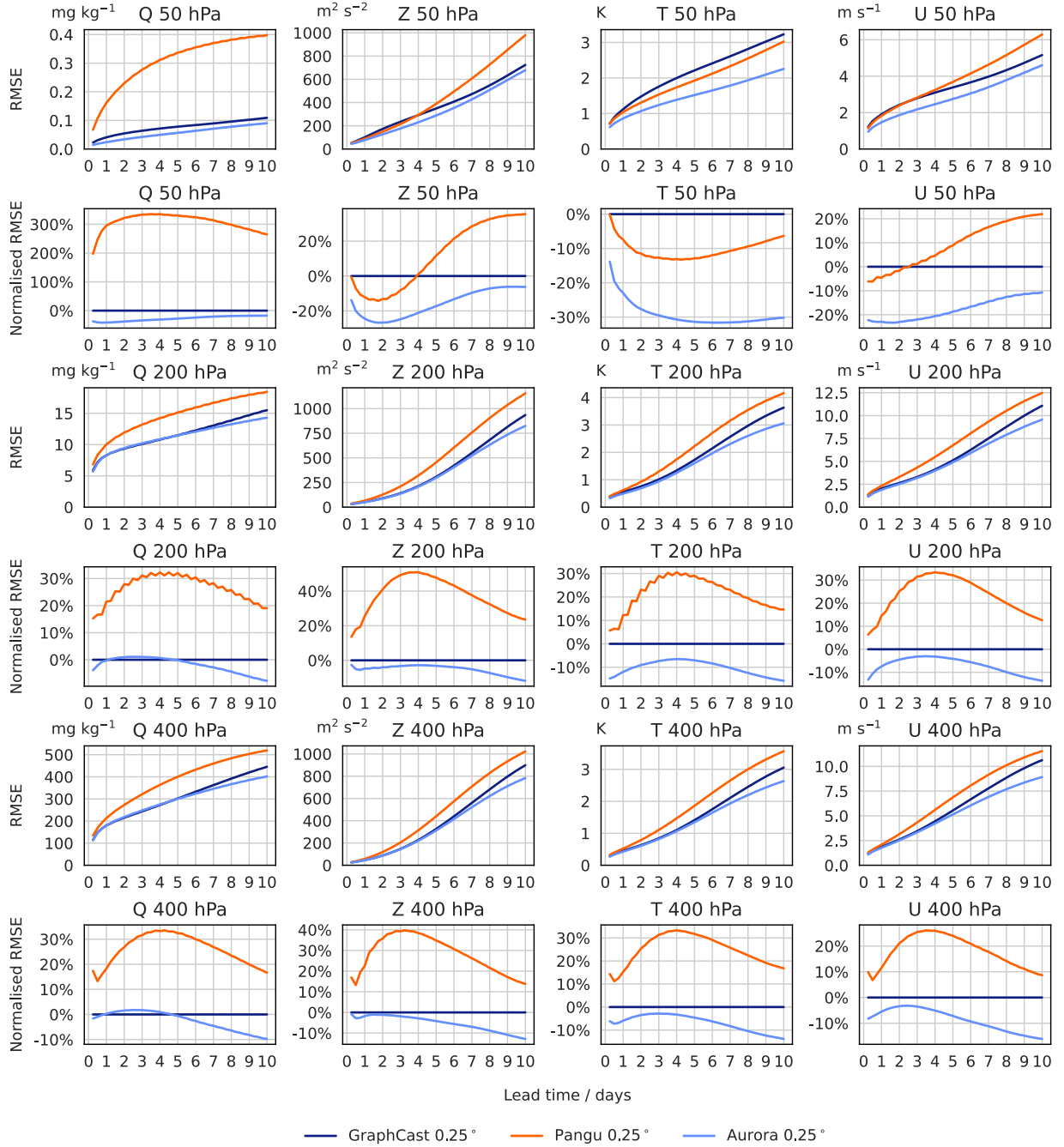


Fig. H9: Root mean square error (RMSE) of Aurora compared to GraphCast in the upper atmosphere. Shows both unnormalised and normalised RMSEs at 0.25° , where GraphCast is used as the baseline. The Pangu weather results at 0.25° are included for reference.

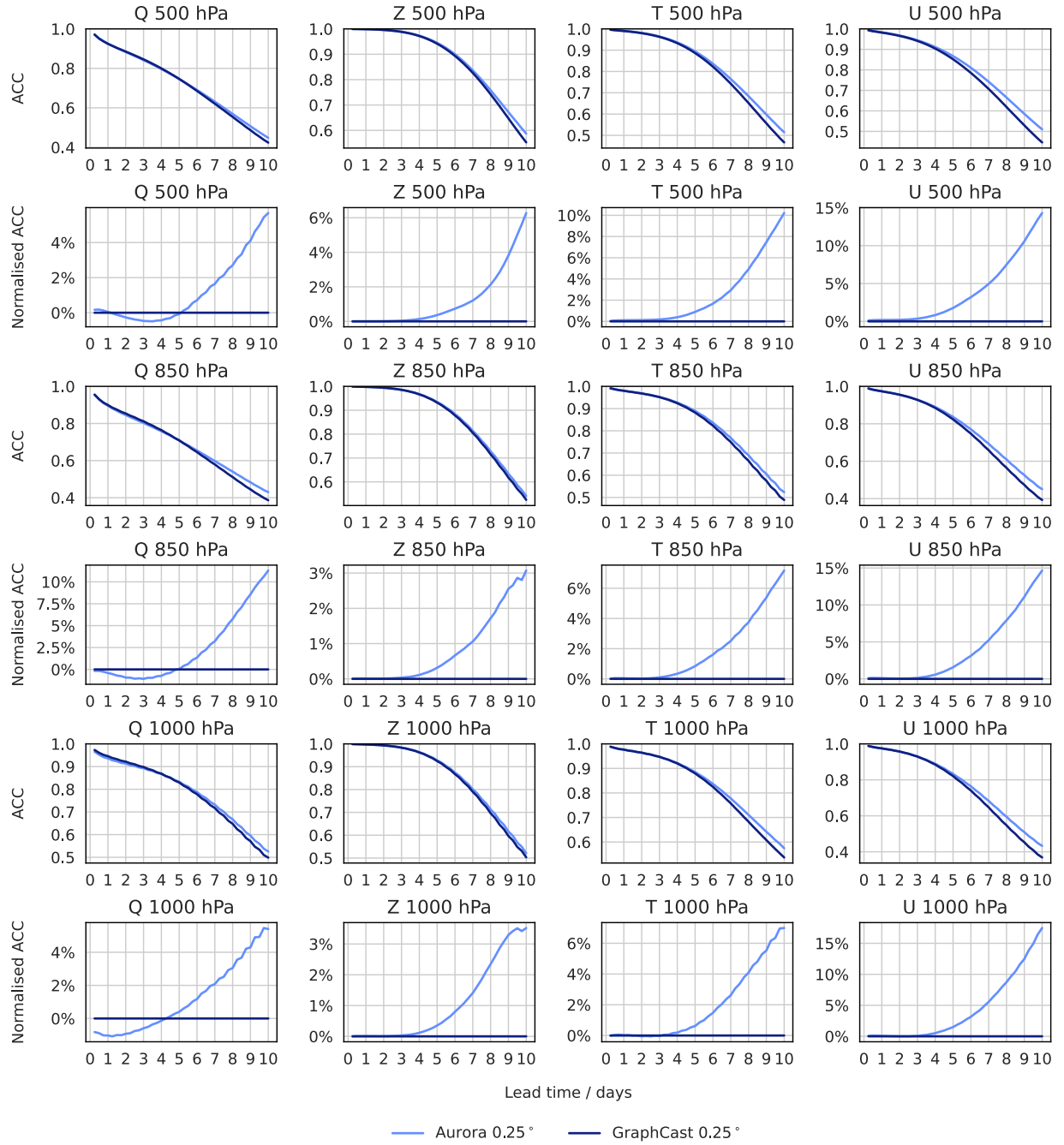


Fig. H10: Anomaly correlation coefficient (ACC) of Aurora compared to GraphCast in the lower atmosphere. Shows both unnormalised and normalised ACCs at 0.25°.

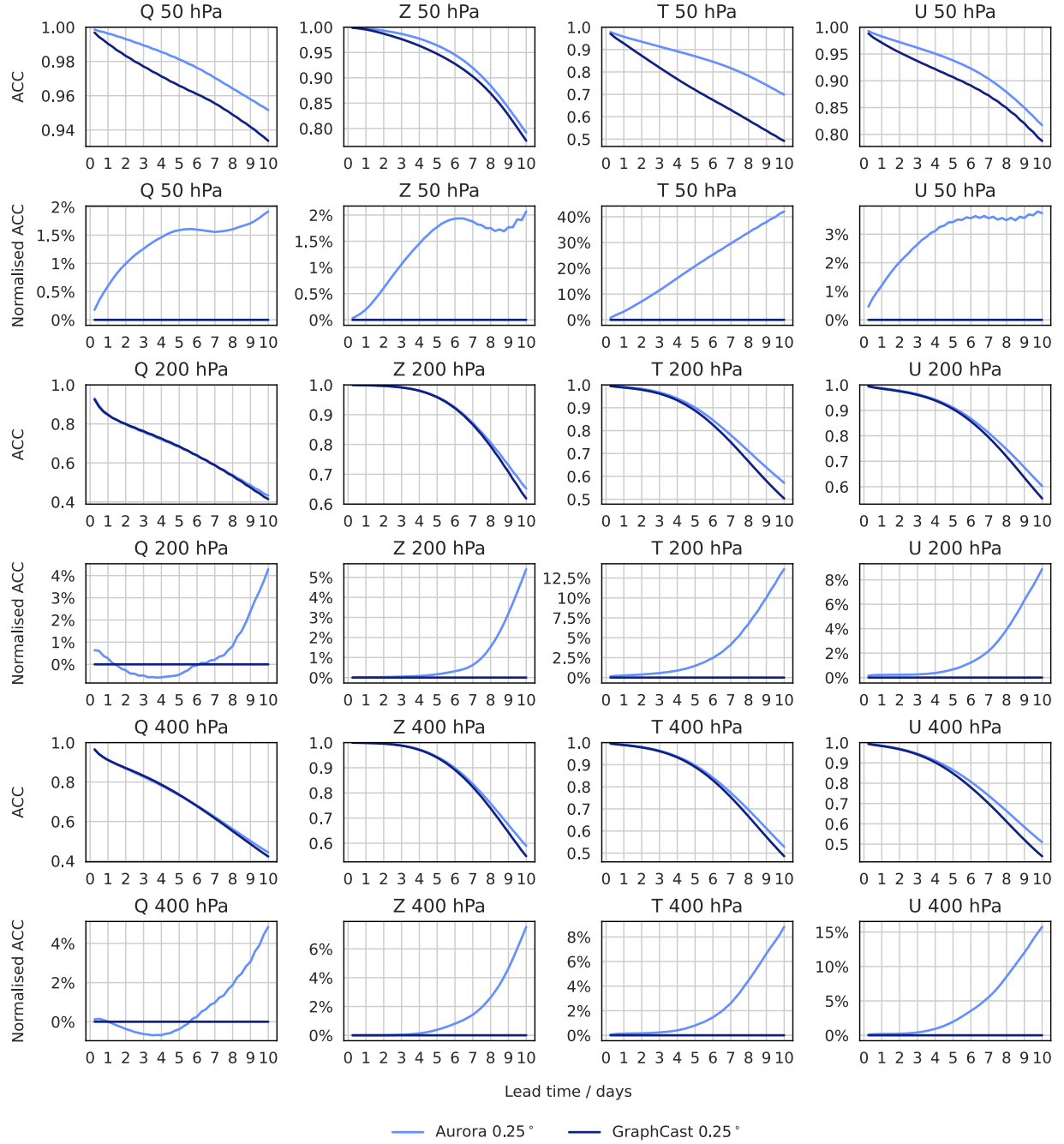


Fig. H11: Anomaly correlation coefficient (ACC) of Aurora compared to GraphCast in the upper atmosphere. Shows both unnormalised and normalised ACCs at 0.25°.

H.3 Aurora versus Pangu at 0.25°

To further expand upon the evaluation of Aurora at 0.25° against existent AI models, we include a scorecard comparison with Pangu-Weather^[2] 0.25° in Figure H12. Aurora consistently out-performs Pangu across all variables and pressure levels.

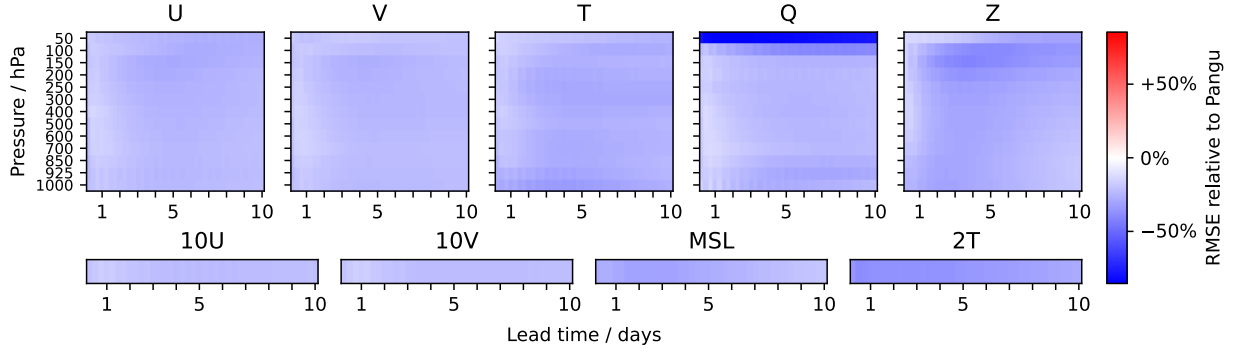


Fig. H12: Scorecard comparing Aurora versus Pangu-Weather. Aurora at 0.25° out-performs Pangu-Weather across all variables and pressure levels.

H.4 Aurora versus IFS HRES at 0.25°

To further expand upon the evaluation of Aurora at 0.25°, we include a scorecard comparison with IFS HRES 0.25° in Figure H13. Aurora largely outperforms HRES, but despite its improvements in the upper atmosphere compared to GraphCast, it still performs worse than HRES at the top of the atmosphere. Figure H14 shows a scorecard comparing Aurora at 0.25° to the IFS ensemble mean. Aurora outperforms the ensemble mean up to 7 days lead time.

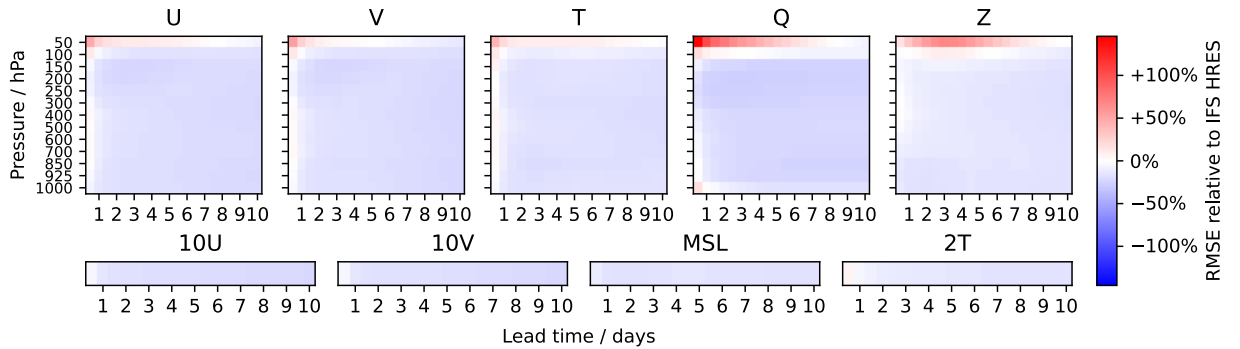


Fig. H13: Scorecard comparing Aurora versus IFS HRES at 0.25°. Despite significantly improving versus GraphCast in the higher atmosphere, Aurora is also worse than HRES at the top of the atmosphere. However, Aurora generally outperforms HRES on other levels.

H.5 Verification against weather station measurements at 0.25°

As for the high-resolution model, we evaluate Aurora 0.25° forecasts issued at 00 UTC between 2 January 2022 and 19 December 2022 against station observations from the WeatherReal-ISD dataset^[46]. Figure H15 shows the RMSE and MAE as a function of lead time for 0.25° AIWP models versus IFS HRES. The relative error in temperature between Aurora and GraphCast tracks very closely with the scorecard results in Figure H6. Despite an advantage in resolution, IFS HRES only leads within the first 12 hours. For wind speed, Aurora outperforms GraphCast by a larger margin over the first 6 days compared to evaluations on gridded data, but performs worse than GraphCast after day 7 on RMSE values, particularly on 18 UTC timestamps. These results show that AIWP models such as Aurora are still capable of achieving high forecast accuracy as measured by observations, even at the coarser 0.25° resolution.

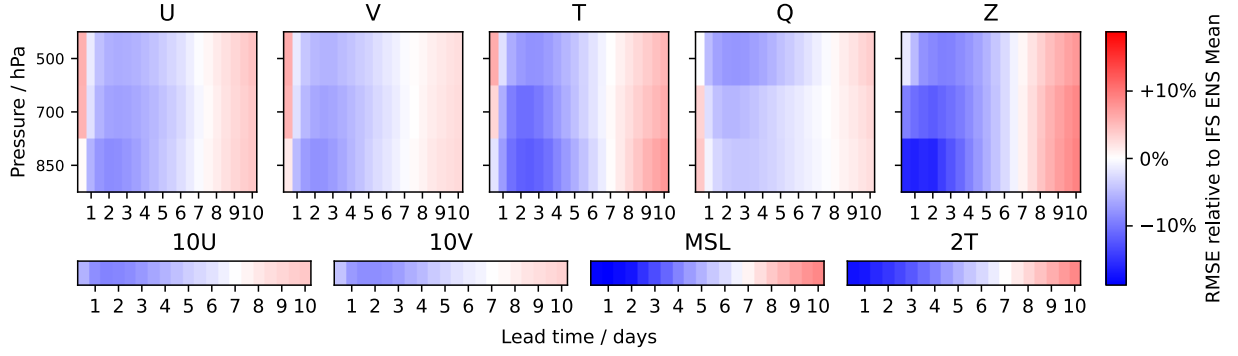


Fig. H14: Scorecard of Aurora versus IFS ENS mean at 0.25° . Aurora outperforms the ensemble mean up to around 7 days lead time.

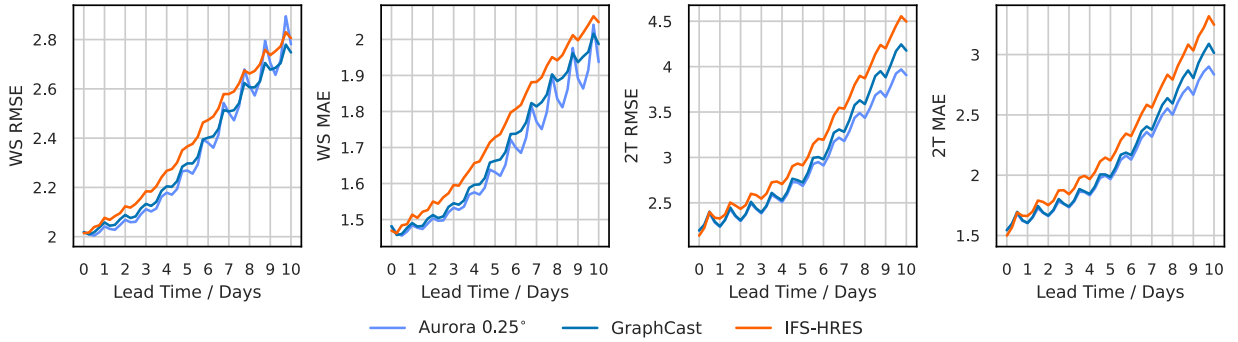


Fig. H15: Evaluation of Aurora, GraphCast and IFS on weather station observations. The panels display the Root mean square error (RMSE) and mean absolute error (MAE) against global weather stations during 2022 for wind speed (left two panels) and surface temperature (right two panels).

H.6 Extreme event prediction at 0.25°

From a meteorological perspective, accurate forecasting of the extremes of surface weather, such as wind speed and temperature, is of critical importance in forward-planning to mitigate the impact on life. To verify the performance of Aurora in predicting the tails of the relevant distributions, we present a comparison with GraphCast and IFS HRES using the 06 UTC and 18 UTC initialisations from the 2022 IFS HRES 0.25° dataset in Figure H16. Here we show RMSEs computed in the upper and lower tails of the data distribution as described in Supplementary F.2, normalized by the RMSEs from IFS HRES. By sweeping a threshold, we can analyse performance as we move further into the extremes. Values for thresholds above zero are shown on the right of the x-axis in each plot, and values for thresholds below zero are shown on the left of the x-axis in each plot.

We find that Aurora outperforms GraphCast on wind speed prediction at the surface. At longer lead times, the performance of AIWP models relative to IFS HRES becomes markedly improved across the distribution. Note however that 2T exhibits a difference in behaviour between the warmer and colder sections of the distribution; it is reported that IFS HRES performance for winter extremes is biased, resulting in less accurate forecasting during winter months^[45]. Since AIWP models include such biased forecasting data rather than just forward simulations within physical constraints, there is a tendency to exhibit these biases when compared to NWP models such as IFS HRES.

In Figure H17, we show the thresholded RMSEs for atmospheric variables distributed over four pressure levels throughout the atmosphere. At the higher pressure levels (i.e., near the surface), results are consistent with those shown for the surface variables in Supplementary H.6. However, the temperature in the atmosphere is more typically distributed as it is less susceptible to bias through near-surface effects. In the higher atmosphere, overall performance is more challenging, although Aurora remains able to outperform GraphCast, consistent with total RMSE results shown in Figure H6.

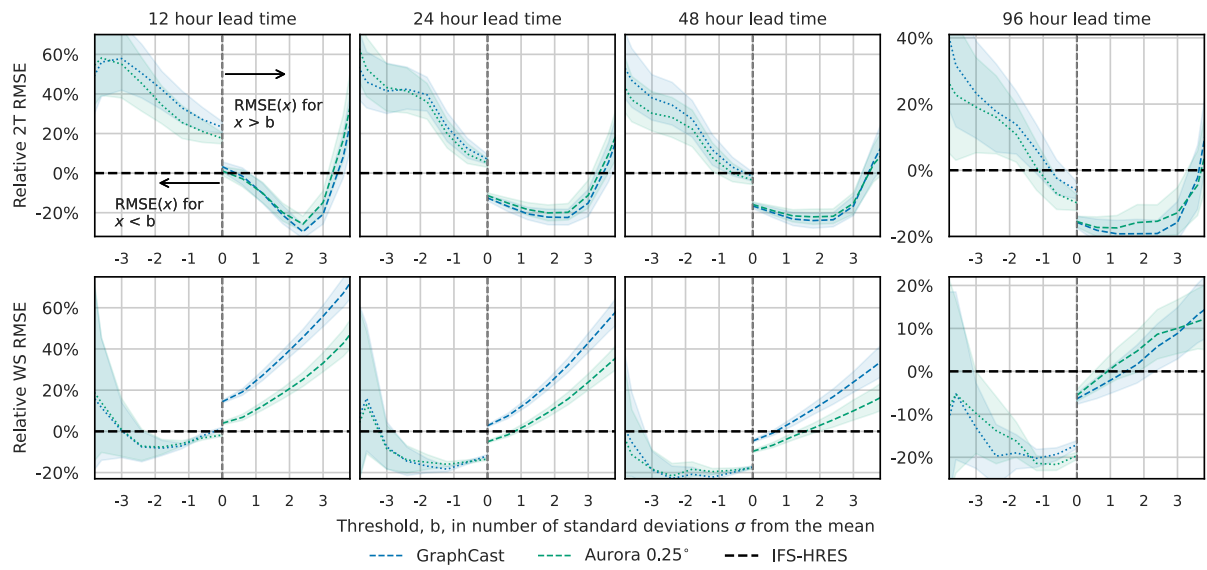


Fig. H16: Thresholded RMSE for Aurora, GraphCast and IFS HRES normalized by IFS HRES performance. Aurora demonstrates improved prediction for the extreme values (i.e., tails) of the surface variable distributions. In each plot, values to the right of the centre line are cumulative RMSEs for targets found to sit above the threshold, while those to the left represent target values sitting below the threshold. The full procedure is outlined in Supplementary F.2. The shading represents 95% confidence intervals obtained through bootstrapping. Specifically, for each threshold value, we compute RMSE values using Equation (F17) in Supplementary F.2, and these values are sampled with replacement to estimate the confidence intervals.

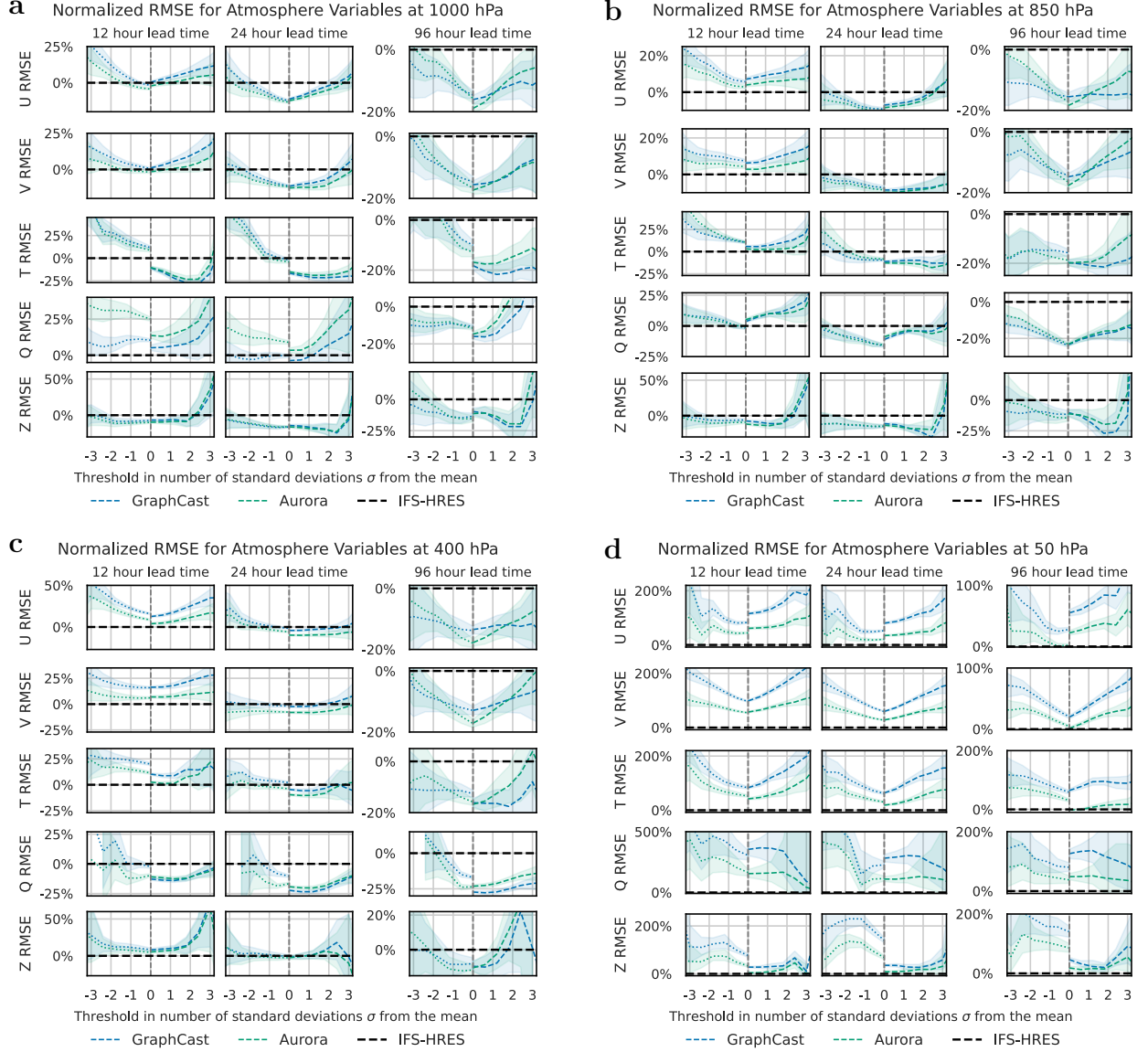


Fig. H17: Thresholded RMSE results for atmospheric variables at four pressure levels distributed throughout the atmosphere, presented as percentages relative to the performance of IFS HRES. All comparisons are at 0.25° resolution. The origin of these curves is explained in Supplementary F.2. The discontinuity at zero reflects that only one side of the curve is shown; on the right hand side of each figure we show RMSE_g where target values are above the threshold determined by g , and on the LHS of the figure, RMSE_g is the cumulative RMSE found against target values below the threshold. The shading represents 95% confidence intervals obtained through bootstrapping. Specifically, for each threshold value, we compute RMSE values using Equation (F17) in Supplementary F.2, and these values are sampled with replacement to estimate the confidence intervals. **a:** Errors at level 1000 hPa. **b:** Errors at level 850 hPa. **c:** Errors at level 400 hPa. **d:** Errors at level 50 hPa.

I Additional results

In this section, we present results that supplement the results in the main body.

I.1 Comparison against operational CAMS

It is important to note that we take CAMS analysis data as the ground truth. This is because CAMS analysis is produced by combining CAMS forecasts with observations (i.e., assimilation), the lack of real-time emission data means that CAMS analysis is more biased towards similarity with CAMS forecasts than is the case for meteorological variables, especially for near-surface pollution variables at short lead times. For example, the PM values are only weakly constrained via aerosol optic depth satellite measurements as CAMS does not assimilate any real-time emission observations. This may disadvantage Aurora in this comparison. Below are the details of the various experiments we conduct.

Figure I18 shows a scorecard for the full collection of CAMS variables. Aurora is competitive to CAMS (within 20% RMSE) on 96% of all targets, and Aurora matches or outperforms CAMS on 76% of all targets. Figure I19 compares Aurora RMSEs to CAMS for only the surface-level variables, Figure I20 compares the RMSE of Aurora, CAMS, and the persistence prediction at three days lead time. Aurora matches or outperforms CAMS on 89% of all variables and outperforms the persistence prediction for all variables and all lead times.

Iraq sandstorm. Figure I21 visualises predictions for PM_{10} by Aurora and CAMS for the sandstorm that hit Iraq on 13 June 2022.

Classification of levels of $\text{PM}_{2.5}$ and PM_{10} . Figure I22 compares the accuracy, precision, and recall of CAMS and Aurora for classifying levels of $\text{PM}_{2.5}$ and PM_{10} according to the categories of AirNow^[26]. Aurora and CAMS both generally obtain very high accuracy, although Aurora predictions come at a fraction of the cost. At longer lead times, Aurora tends to obtain higher precision and lower recall, meaning that predictions by Aurora are generally more conservative. For example, for unhealthy (or worse) levels of $\text{PM}_{2.5}$, on average Aurora has 4% higher precision but 8% lower recall.

Spectral analysis. Figure I23 includes a spectral analysis of Aurora’s predictions for all surface-level variables, performed according to the method in Lam et al.^[3]. For longer lead times, power at smaller wave lengths tends to decrease, meaning that the predictions tend to become more smooth. For TC NO, TC CO, TC NO₂, TC SO₂, and TC O₃, this decrease is comparable to that for the meteorological variables 2T, U10, V10, and MSL. For PM_1 , $\text{PM}_{2.5}$, and PM_{10} , the decrease is larger.

Comparison against a model trained from scratch on CAMS data. We train the model from scratch on the CAMS analysis training dataset detailed in Supplementary C.4 to clarify the impact of pretraining on our ability to predict atmospheric variables. The scorecard results are presented in Figure I24. Fine-tuned Aurora consistently outperforms any attempt to train a model to predict these variables from scratch, likely due to the inherently challenging distributional behaviour of these variables, and the small amount of data available.

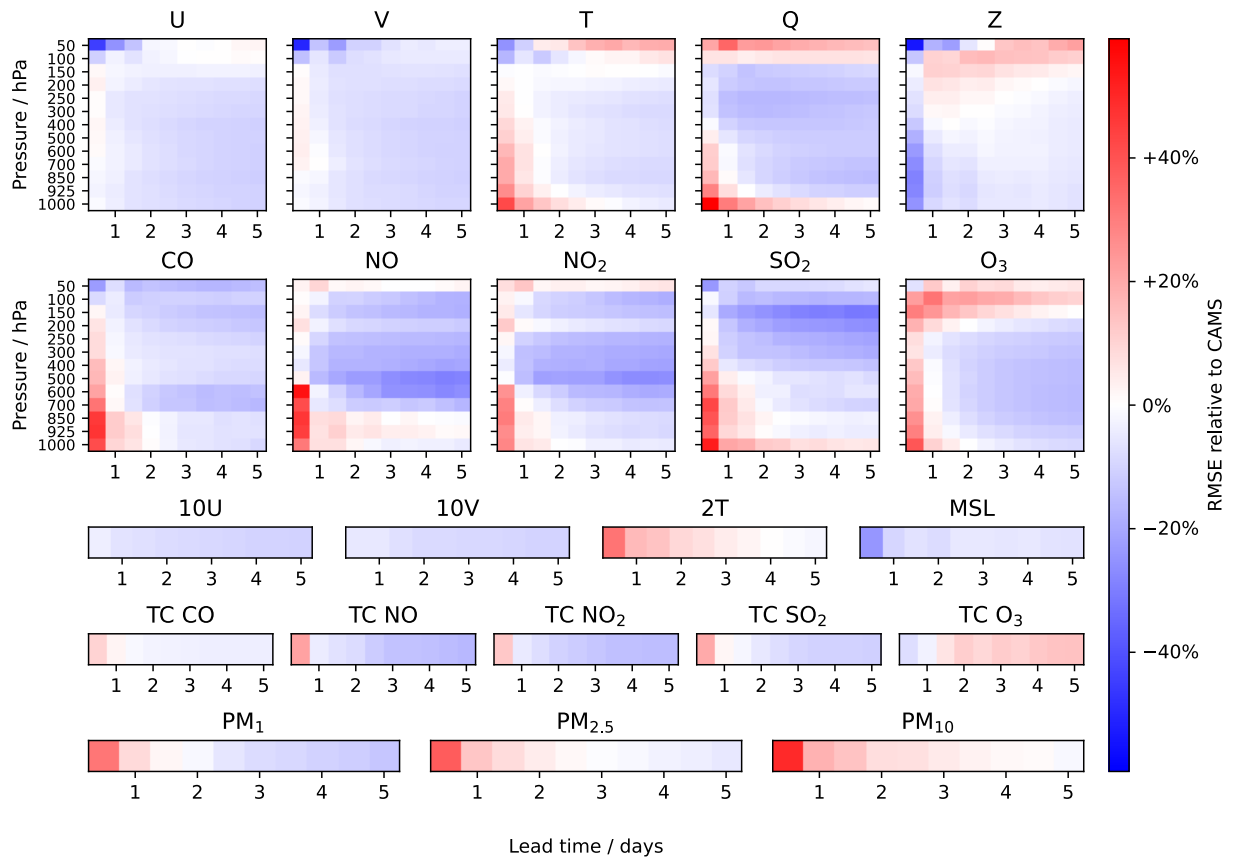


Fig. I18: RMSE-based scorecard comparing Aurora and CAMS. The RMSEs are computed over the period Jun 2022 to Nov 2022 inclusive. Aurora is competitive to CAMS (within 20% RMSE) on 96% of all targets, and Aurora matches or outperforms CAMS on 76% of all targets.

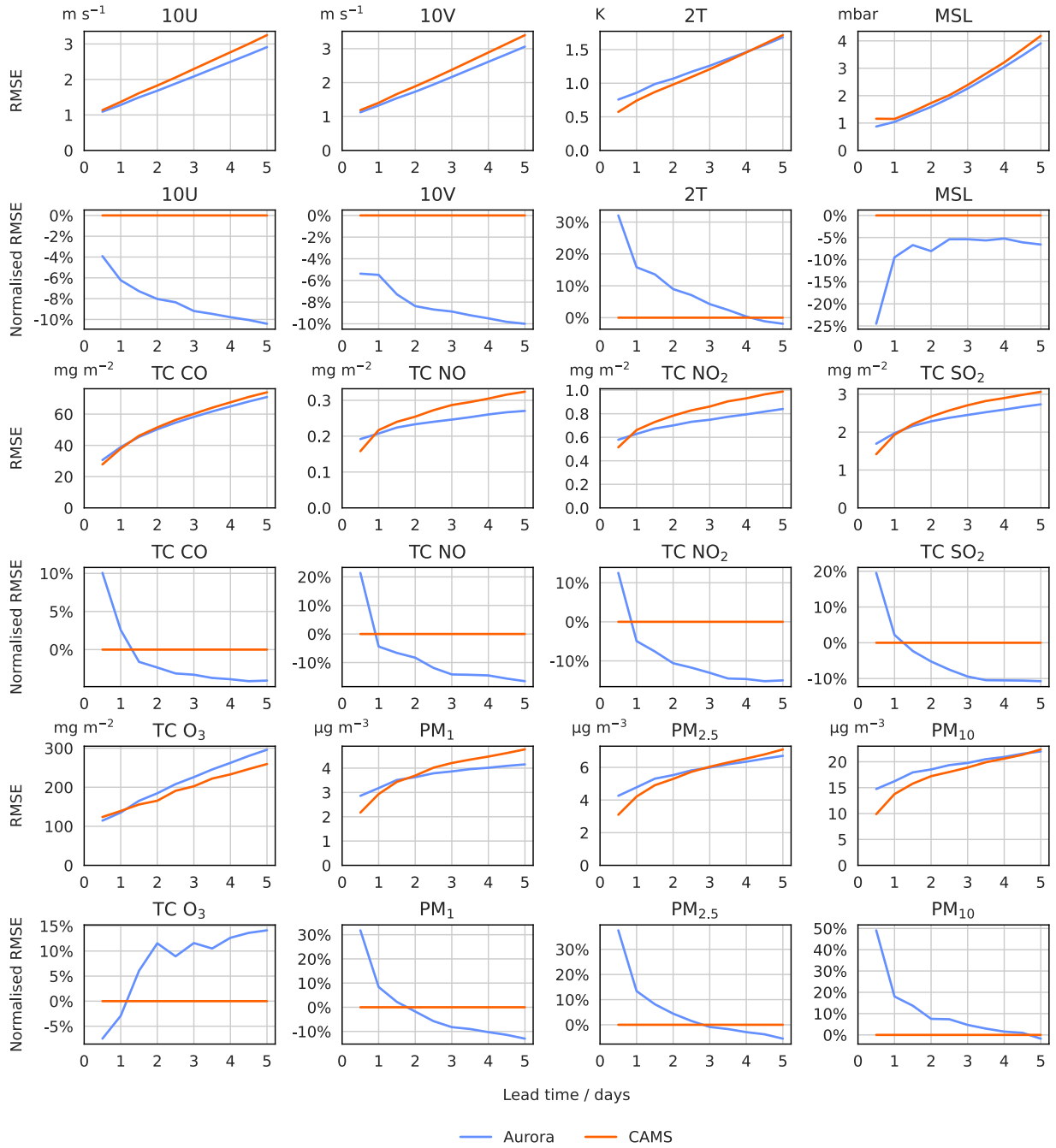


Fig. I19: Latitude-weighted root mean square error (RMSE) of the surface-level variables compared to CAMS. The RMSEs are computed over the period Jun 2022 to Nov 2022 inclusive.

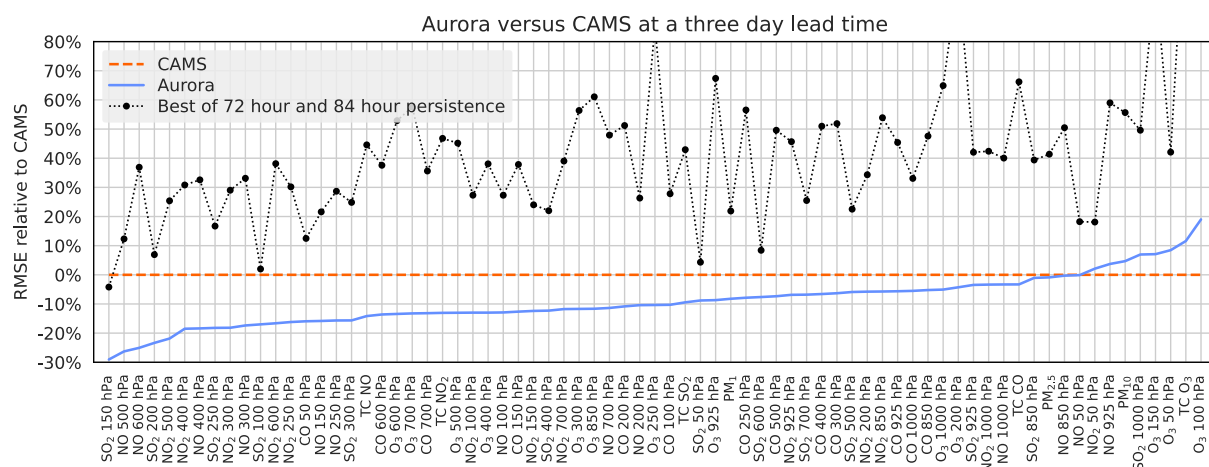


Fig. I20: Latitude-weighted root mean square error (RMSE) of Aurora vs CAMS and persistence at three days lead time. Negative values mean that Aurora is better. The RMSEs are computed over the period Jun 2022 to Nov 2022 inclusive. Aurora matches or outperforms CAMS on 89% of all variables and is strictly significantly better than the persistence prediction.

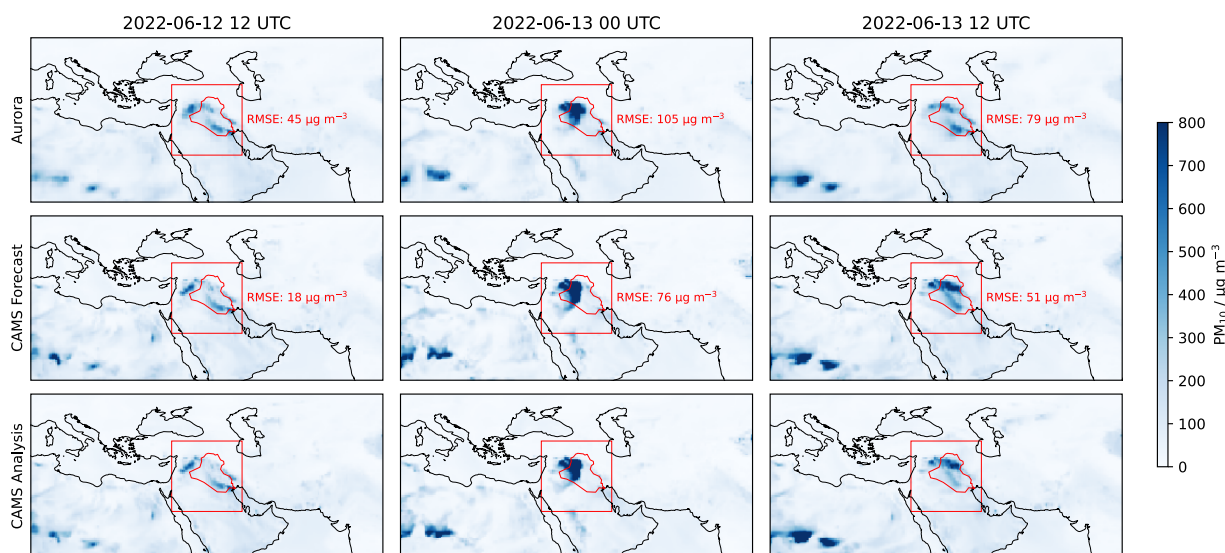


Fig. I21: Predictions for PM₁₀ by Aurora and CAMS for the sandstorm that hit Iraq on 13 June 2022. Aurora and CAMS were initialised with CAMS analysis at 12 Jun 2022 00 UTC. Aurora and CAMS both predict the sandstorm (highlighted by the red box) a day in advance. The figure also shows the RMSE computed over the red rectangle.

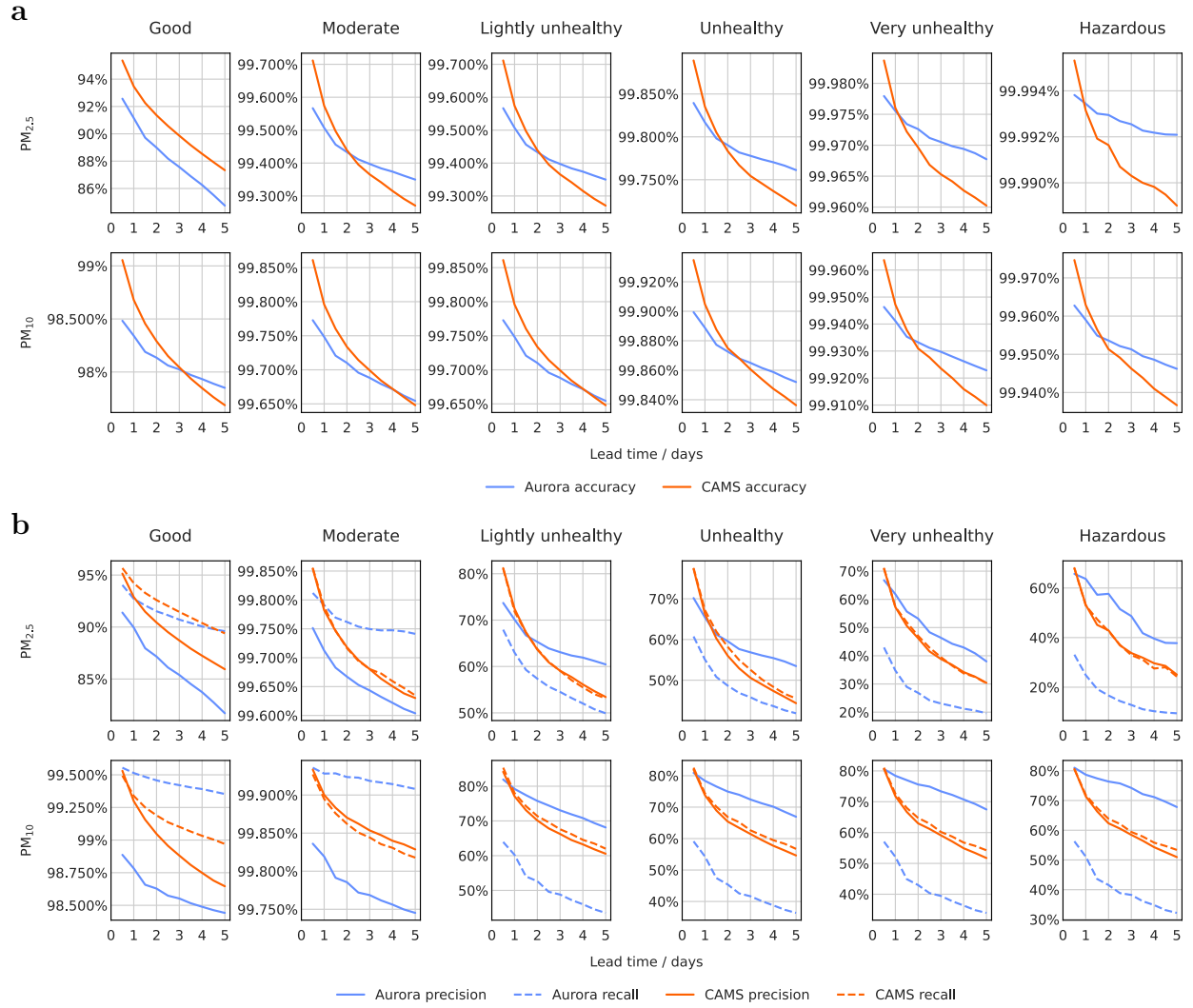


Fig. I22: Comparison of Aurora and CAMS for classifying levels of $\text{PM}_{2.5}$ and PM_{10} according to the categories from AirNow. **a:** Classification accuracy of levels of $\text{PM}_{2.5}$ according to the following categories: good ($9.1 \mu\text{g m}^{-3}$ or lower), moderate ($35 \mu\text{g m}^{-3}$ or lower), lightly unhealthy ($35 \mu\text{g m}^{-3}$ or higher), unhealthy ($55 \mu\text{g m}^{-3}$ or higher), very unhealthy ($125 \mu\text{g m}^{-3}$ or higher), and hazardous ($225 \mu\text{g m}^{-3}$ or higher). Also shows the classification accuracy of levels of PM_{10} according to the same categories: good ($55 \mu\text{g m}^{-3}$ or lower), moderate ($155 \mu\text{g m}^{-3}$ or higher), lightly unhealthy ($155 \mu\text{g m}^{-3}$ or higher), unhealthy ($255 \mu\text{g m}^{-3}$ or higher), very unhealthy ($355 \mu\text{g m}^{-3}$ or higher), and hazardous ($425 \mu\text{g m}^{-3}$ or higher). The accuracies are computed over the period Jun 2022 to Nov 2022 inclusive. **b:** Like **a**, but shows precision and recall.

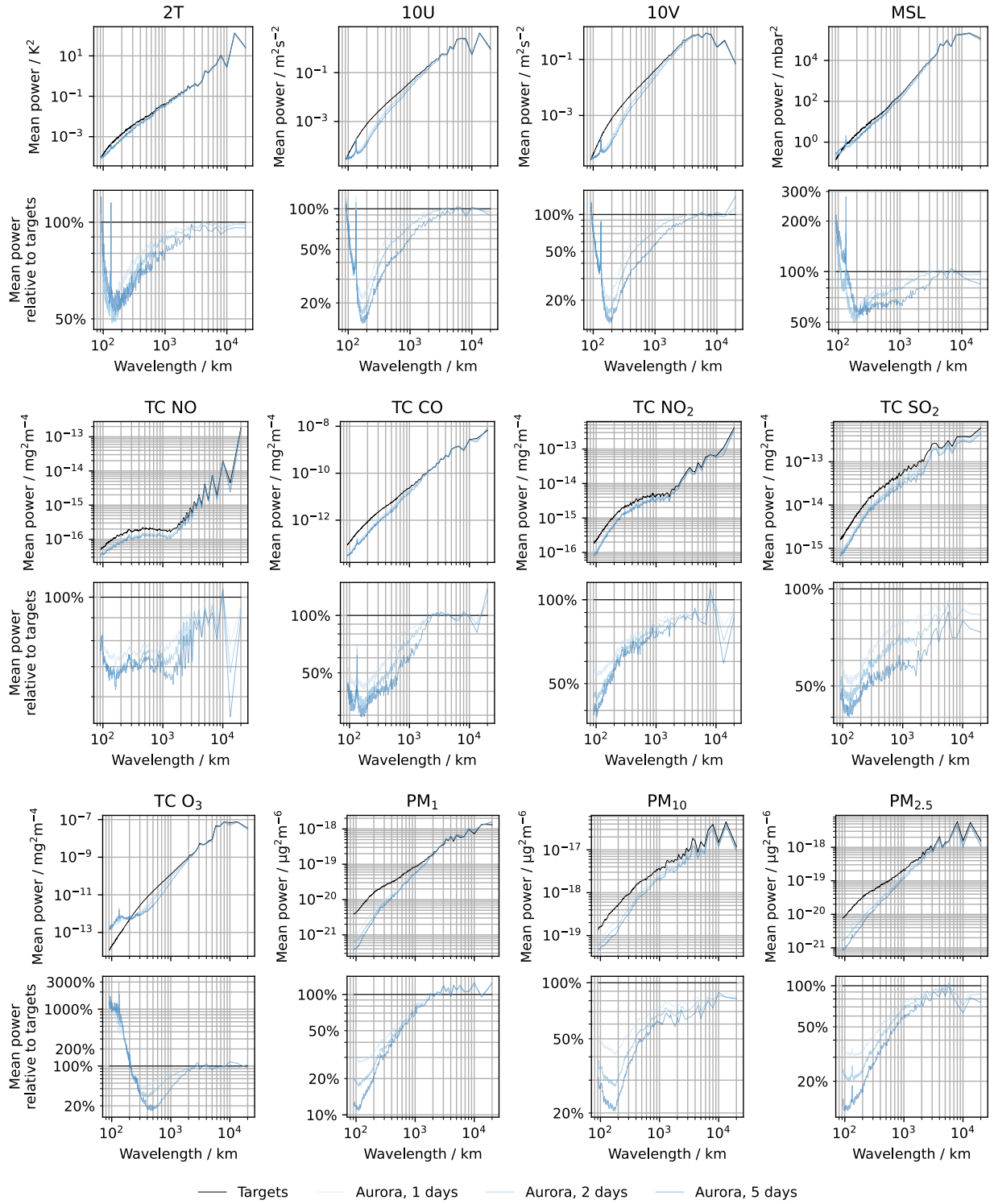


Fig. I23: Spectral analysis of the predictions for the surface-level variables by Aurora, compared to CAMS analysis. The computation uses the period Jun 2022 to Nov 2022 inclusive.

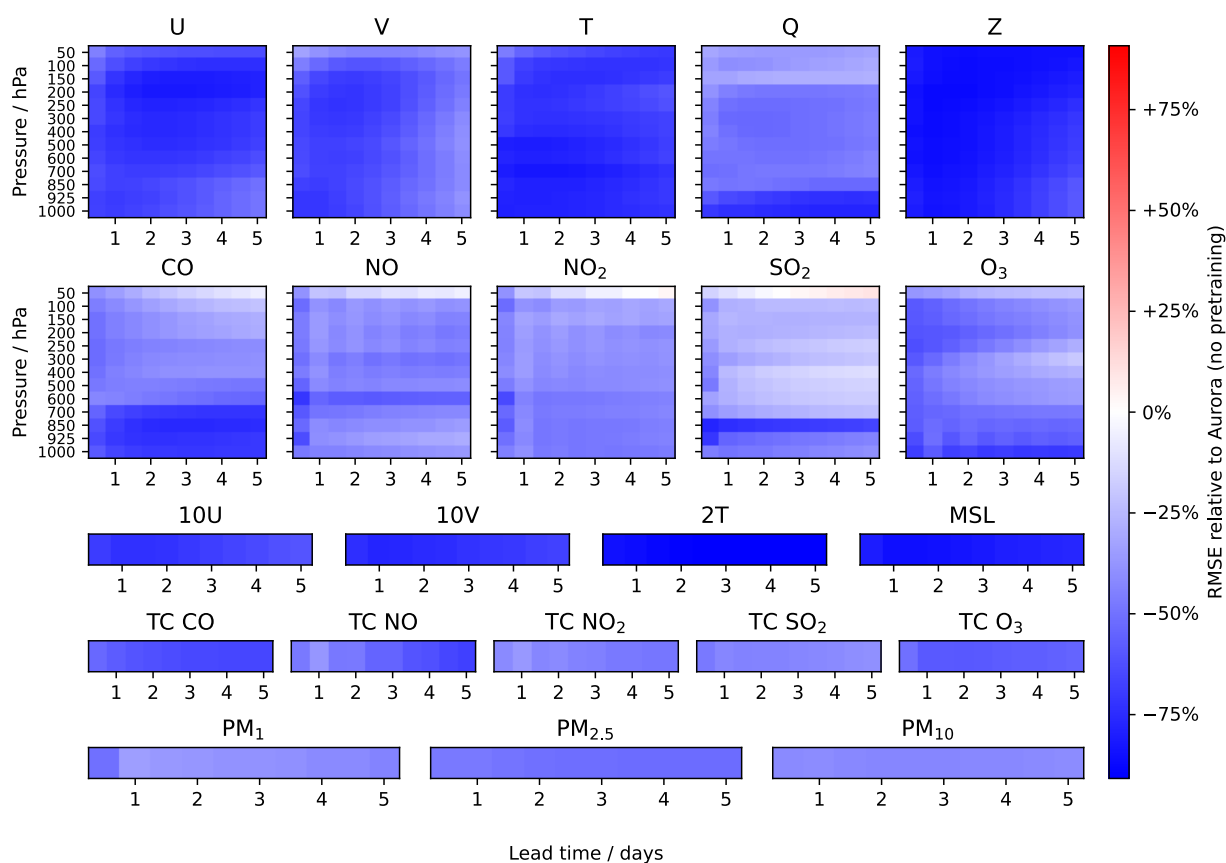


Fig. I24: Scorecard comparing performance of Aurora model fine-tuned on CAMS data to a model trained from scratch on the same data. Fine-tuned Aurora consistently outperforms any attempt to train a model to predict these variables from scratch. On average, pretraining improves performance by 54%.

I.2 Comparison against operational IFS HRES-WAM

Figure I25 shows a scorecard for all wave and meteorological variables. Aurora is competitive (within 20% RMSE) to HRES-WAM on 95% of all targets, and Aurora matches or outperforms HRES-WAM on 85% of all targets. Figures I26 and I27 compare Aurora’s RMSEs to HRES-WAM for all meteorological and wave surface-level variables.

Comparison against a model trained from scratch on HRES-WAM analysis. To investigate the impact of pretraining, we train a model from scratch on HRES-WAM analysis and compare this model to the fine-tuned version of Aurora. The scorecard results are presented in Figure I28. Fine-tuning the pretrained model consistently outperforms the model trained from scratch.

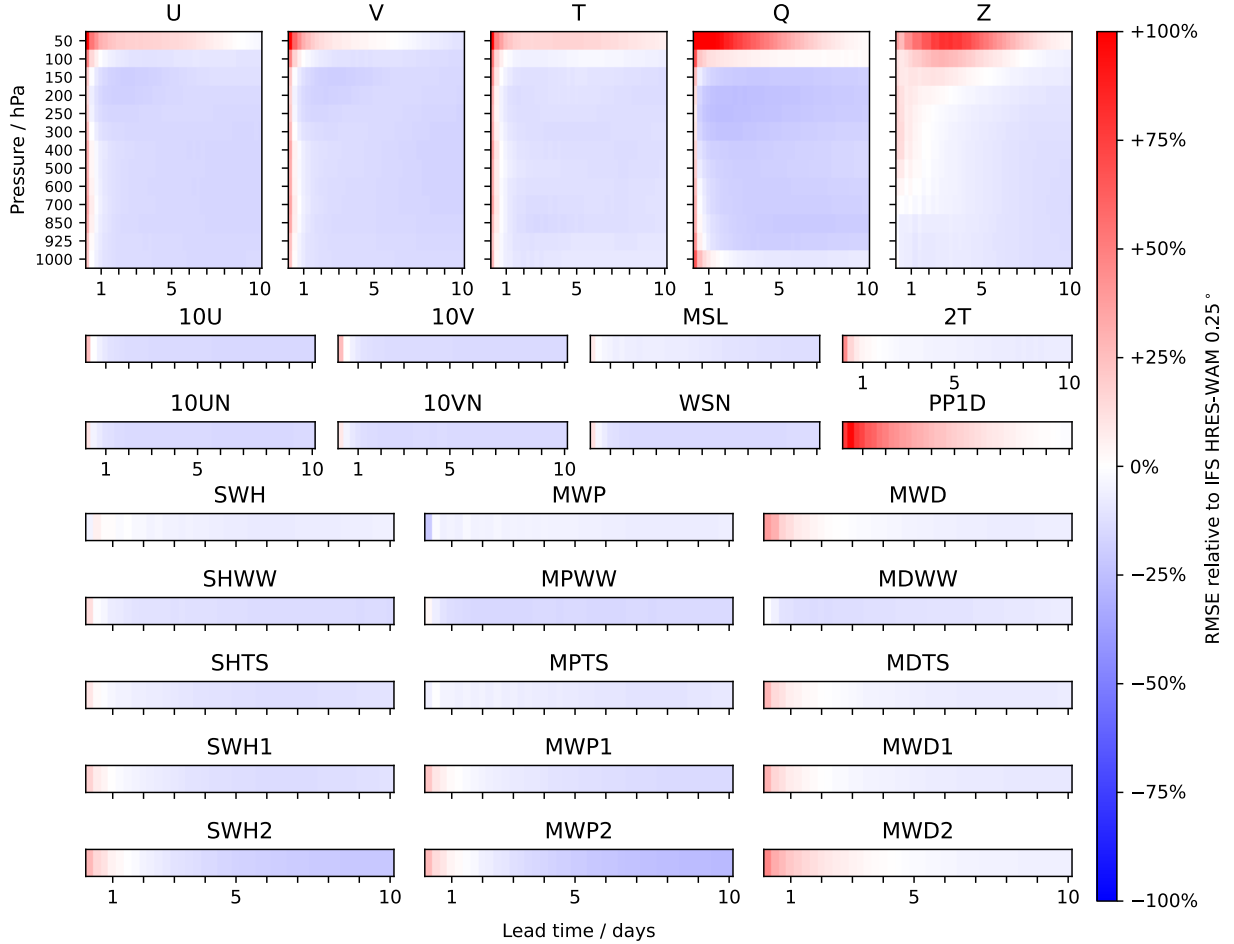


Fig. I25: Latitude-weighted RMSE of Aurora 0.25° relative to HRES-WAM 0.25° Negative values (blue) mean that Aurora is better. Only for Q at 50 hPa, the RMSE initially exceeds 100% and peaks at 306%; no other variable exceeds 100%. Aurora is competitive (within 20% RMSE) to HRES-WAM on 95% of all targets, and Aurora matches or outperforms HRES-WAM on 85% of all targets. Note that the meteorological variables are from HRES T0, and the surface-level wave variables from HRES-WAM. See Section 4 in the main body.

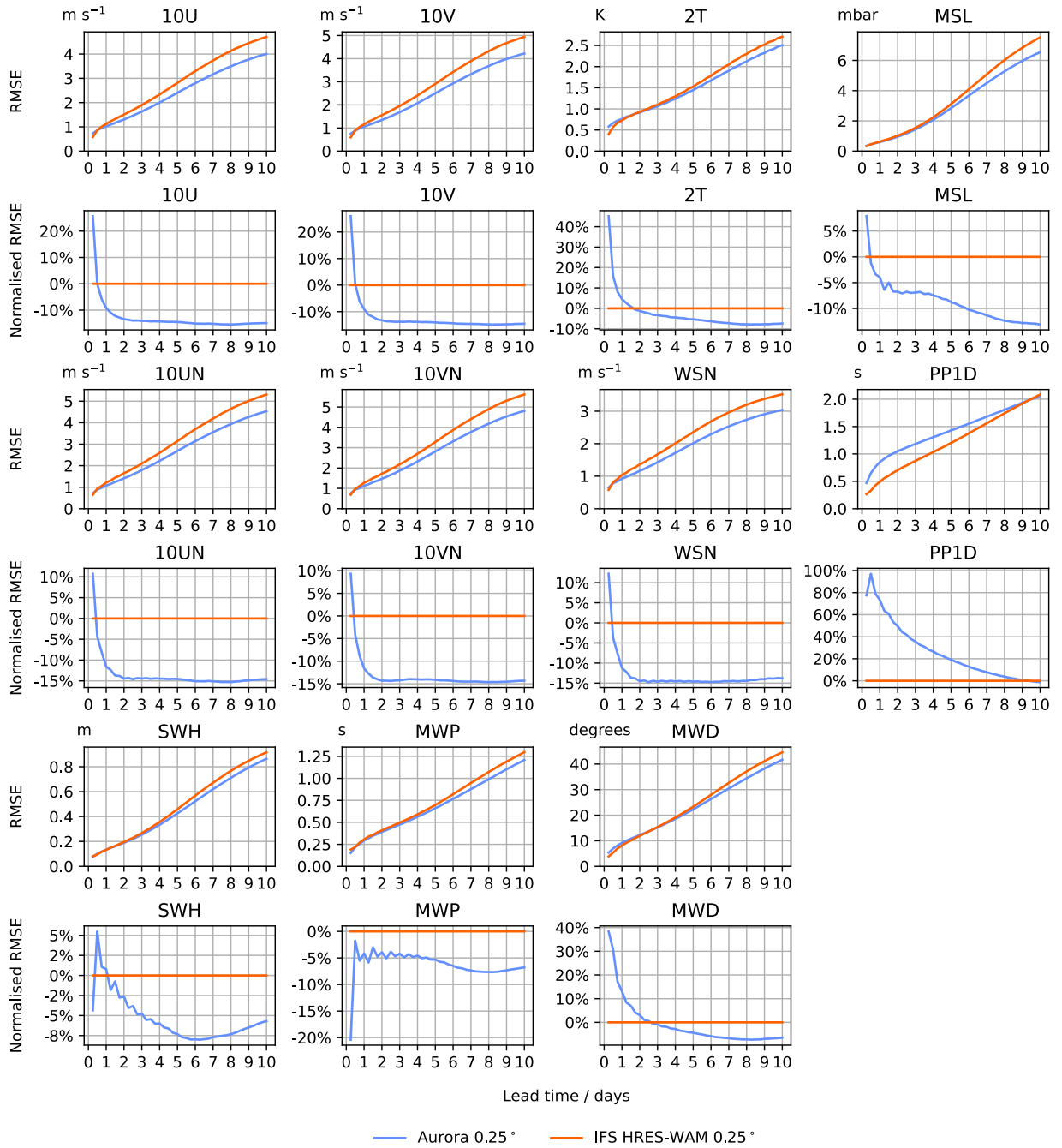


Fig. I26: Latitude-weighted RMSE of Aurora 0.25° compared to HRES-WAM 0.25° on the meteorological surface-level variables and main wave variables. Shows both unnormalised and normalised RMSEs.

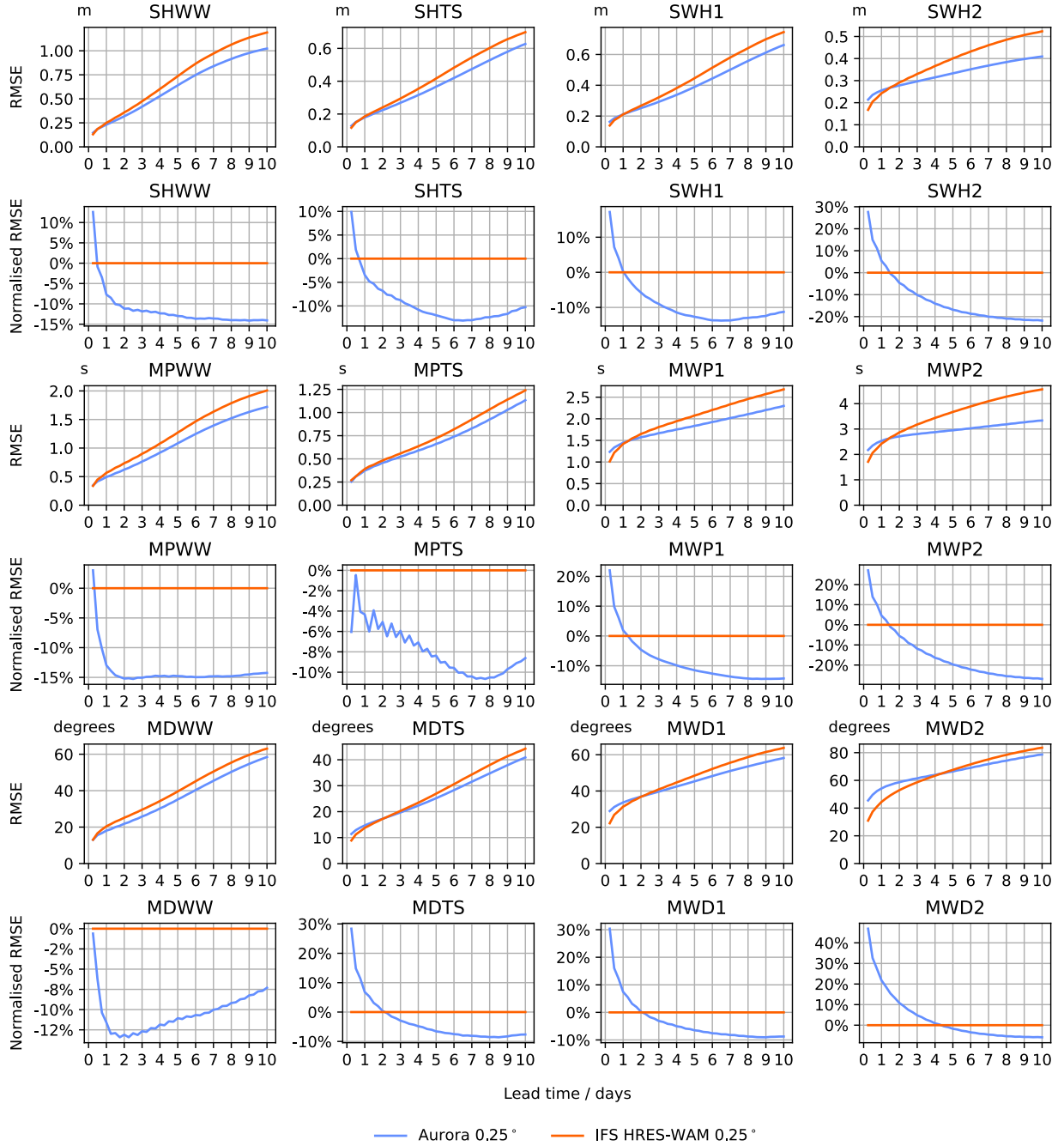


Fig. I27: Latitude-weighted RMSE of Aurora 0.25° compared to HRES-WAM 0.25° on the wave variables of the various wave components. Shows both unnormalised and normalised RMSEs.

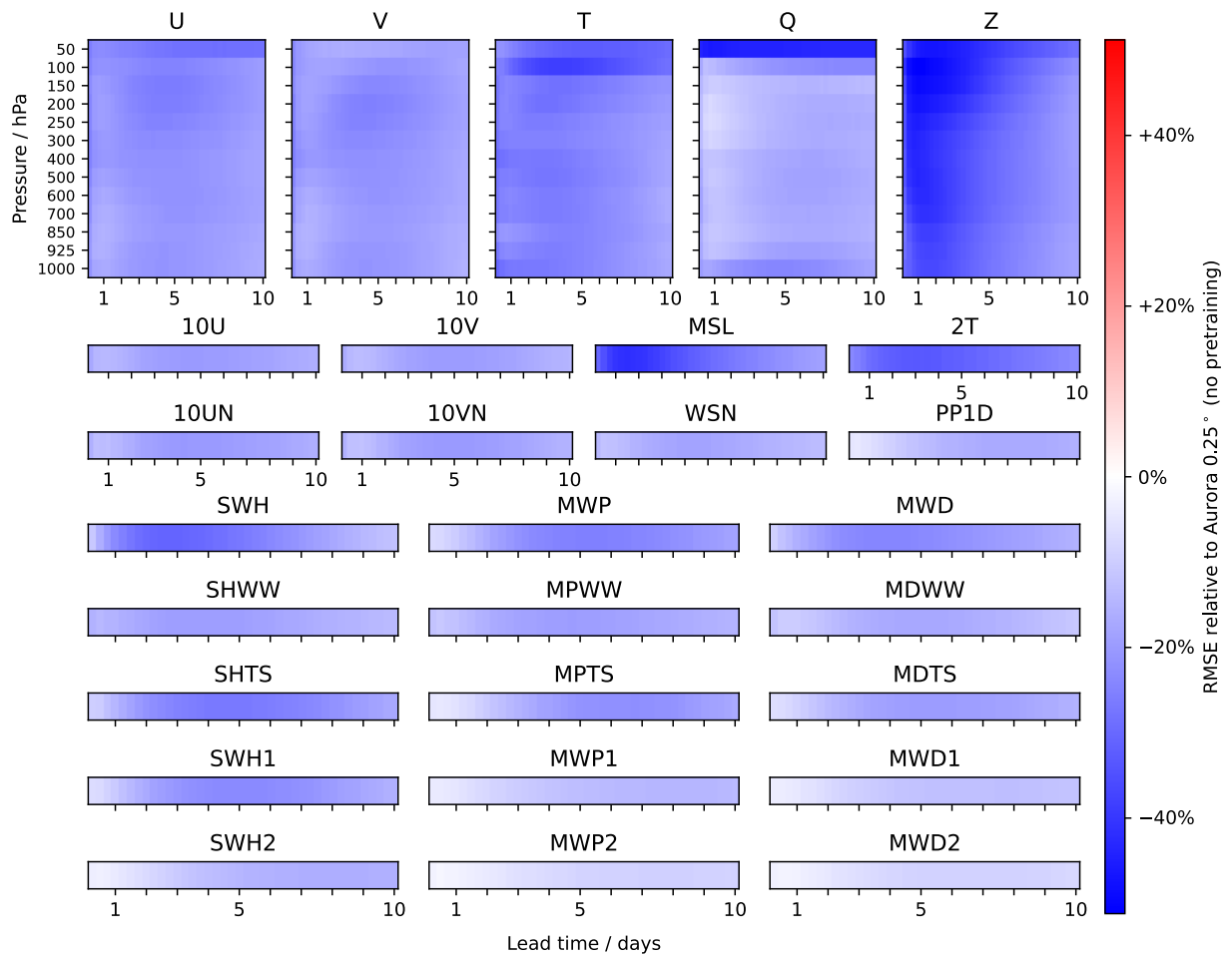


Fig. I28: Scorecard comparing the performance of Aurora fine-tuned on HRES-WAM data to a model trained from scratch (i.e. no pretraining) on the same data. Fine-tuning a pretrained Aurora consistently outperforms the version fine-tuned without pretraining. On average, pretraining improves performance by 22%.

I.3 Details for tropical cyclone forecasting

In this appendix, we describe the tropical cyclone tracker and provide more detail on how the comparison against various agencies in various regions was performed.

I.3.1 Baselines and ground truth

All baseline models from Figure 4a are summarised in Table I6. For the North Atlantic and East Pacific we compare Aurora to both the official forecast and multiple forecast models used by human forecasters to create these products. In all cases we compare the early version of Aurora (described below) to official and early models, ensuring that the comparison is between products available at the forecast time. Data for the official forecasts and all guidance models is taken from the Automated Tropical Cyclone Forecasting System A-Decks^[57]. For the Northwest Pacific, multiple agencies issue official forecasts. We compare Aurora to forecasts from the Joint Typhoon Warning Centre, Central Weather Administration Taiwan, China Meteorological Administration, Japan Meteorological Agency and Korea Meteorological Administration. Finally, for the Australian Region, we compare to the official predictions from the Australian Bureau of Meteorology.

For ground truth we use the International Best Track Archive for Climate Stewardship (IBTrACS). IBTrACS is a global database that consolidates tropical cyclone data from multiple meteorological agencies worldwide. It provides comprehensive historical records on storm tracks, intensities, and characteristics, supporting climate research and storm impact analysis.

I.3.2 Considerations for fair comparison to baselines

In making comparisons to the baselines, we are careful not to advantage Aurora over the baselines in any way. Two key considerations need to be made: dealing with early versus late models and instances where the tracker fails.

Early versus late models. In tropical cyclone forecasting, models are classified as either early or late depending on when they become available in the forecast cycle. Track forecast models are initialised at UTC 00, UTC 06, UTC 12 and UTC 18. Official track forecasts are due three hours after these times. *Early models* are models whose predictions are available before the end of this three hour window. Predictions of early models can therefore be used to guide the official prediction. In contrast, predictions of *late models* become available after the three hour window. Predictions of late models cannot be used to guide the official prediction. To use late models to guide the official prediction, predictions from one forecast cycle earlier must be used. For example, the prediction for the initialisation at UTC 06 can be used guide the official track prediction at UTC 12.

Although Aurora is able to be run in minutes, the assimilation window for the initial conditions used in this study, HRES T0, is up to three hours past the analysis time. As this places Aurora right on the divide between early and late models, out of an abundance of caution we opt to delay all predictions by six hours. For example, the UTC 12 track prediction from Aurora is generated by running Aurora from the HRES T0 initial conditions from UTC 06. This ensures that Aurora is an early model and suitable for comparison with the official forecasts.

Tracker failure. The tracker used in this study is much simpler than those used operationally. Whilst we manually verified that this simple tracker achieves excellent performance in the vast majority of cases, it occasionally fails, resulting in incorrect predictions. To not advantage Aurora in any way, we never reject any track prediction, meaning that we also retain the track prediction whenever the tracker fails. In fact, this slightly disadvantages Aurora in the comparison, as the baselines and IBTrACS use state-of-the-art trackers.

I.3.3 Tracker

Our heuristic tracker is based on forecasts by Aurora 0.25° (Supplementary H) initialised with HRES T0. The tracker attempts to follow a local minimum in predictions for MSL. A challenge in this approach is that MSL may be plagued by multiple local minima due to local topological features like mountains. We therefore adopt a multi-step procedure that resorts to Z700 when detection of the MSL local minimum fails.

When we say that the tracker attempts to find the local minimum closest to a reference position in a $x^\circ \times x^\circ$ box, the tracker performs the following procedure. It first selects a $x^\circ \times x^\circ$ box centred around the reference

Table I6: Baselines for the tropical cyclone tracking experiment. NHC refers to National Hurricane Centre.

Model	Agency	Description
NORTH ATLANTIC AND EAST PACIFIC		
OFCL	NHC	Official forecast
OCD5	NHC	Statistical baseline: climatology and persistence model
HWFI	NHC	Multi-layer regional dynamical: Hurricane Weather and Research Forecasting Model
HMNI	NHC	Multi-layer regional dynamical: Hurricanes in a Multi-Scale Ocean-Coupled Non-Hydrostatic Model
CTCI	NHC	Multi-layer regional dynamical: COAMPS-TC using GFS initial and boundary conditions
NVGI	NHC	Multi-layer global dynamical: Navy Global Environmental Model
EMXI	NHC	Multi-layer global dynamical: ECMWF global model
CMCI	NHC	Multi-layer global dynamical: Environment Canada global model
AEMI	NHC	Consensus: GFS ensemble mean
FSSE	NHC	Corrected consensus: FSU super-ensemble
TVCA	NHC	Consensus: average of at least two of GFSI, EGRI, HFAI, HFBI, EMXI, CTCI, HWFI HMNI EMNI
HCCA	NHC	Corrected consensus: weighted average of AEMI, CTCI, EGRI, EMNI, EMXI, GFSI, HFAI, HFBI, HWFI, and UEMI
NORTHWEST PACIFIC		
PGTW	Joint Typhoon Warning Centre	Official forecast
CWA	Central Weather Administration Taiwan	Official forecast
BABJ	China Meteorological Administration	Official forecast
RJTD	Japan Meteorological Agency	Official forecast
RKSL	Korea Meteorological Administration	Official forecast
AUSTRALIAN REGION (AUS)		
BoM	Australian Bureau of Meteorology	Official forecast

position. Within this box, a point is classified as a local minimum if it attains the lowest value within a further $2^\circ \times 2^\circ$ box centred around that point. The tracker then finds the local minimum (according to this classification) in the $x^\circ \times x^\circ$ box closest to the reference position. If this closest local minimum lies on the edge of the $x^\circ \times x^\circ$ box, then we consider this a failure and say that no local minimum can be found.

The tracker is initialised with the current latitude–longitude position of the TC. Let the *current position* be the initialisation. If a $5^\circ \times 5^\circ$ box around the current position is clear of land, the tracker attempts to update the current position to the closest local minimum in this box for the 6 h prediction of MSL. If the current position was not updated, the tracker attempts a $4^\circ \times 4^\circ$ box, a $3^\circ \times 3^\circ$ box, a $2^\circ \times 2^\circ$ box, and eventually a $1.5^\circ \times 1.5^\circ$ box. If the current position is then still not updated, the tracker attempts to update the current position to the closest local minimum in a $5^\circ \times 5^\circ$ box for the 6 h prediction of Z700, and then repeats the procedure with varying box sizes for MSL. The resulting current position is the track prediction at lead time 6 h.

For all following lead times, a guess $\tilde{x}_{t+1}$ is made for the new latitude–longitude position by linearly extrapolating the sequence of tracked latitudes and longitudes based on the previous eight estimates (corresponding to two days). Now call this linear extrapolation the current position. Using predictions 6 h further into the future, the current position is then updated exactly as for the initialisation. The resulting current position is the track prediction 6 h further into the future. We repeat this procedure until a track up to five days lead time has been produced.

I.3.4 Details for Figure 4a

To compute a column in Figure 4a, we consider all TCs in 2022 and 2023 in the relevant region. For every storm, we consider all times when a latitude–longitude position is available in IBTrACS. Call such a time a *forecast time*. From all forecast times, we only consider the times when the associated storm is classified as disturbance or tropical. For every such forecast time, we generate a five-day track forecast with Aurora and consider the track forecast by the model on the x -axis. We then compute the track forecast errors at every lead time where the storm is classified as disturbance or tropical. This restriction to forecast times and lead times where the the storm is classified as disturbance or tropical is conform the NHC evaluation

procedure^[42]. By averaging over all storms and forecast times, we obtain a mean absolute error (MAE) at lead times one to five days for both Aurora and the model on the x -axis. If the model on the x -axis happens to have no prediction at a certain lead time for a particular forecast time, then we ignore that particular lead time for that particular forecast time in the mean computation for both Aurora and the model on the x -axis. We note that this slightly advantages the baselines in this comparison, since Aurora is forced to always produce a forecast. The MAEs at all lead times between Aurora and the model on the x -axis are finally compared to produce the column of Figure 4a.

Call a forecast time for a particular storm a *storm-time* pair. Following the above procedure, for the North Atlantic and East Pacific, we consider 1333 storm-time pairs from 76 different storms; for the West Pacific, 737 storm-time pairs from 47 different storms; and for the Australian region, 213 storm-time pairs from 35 different storms.

Finally, for every square of Figure 4a, we compute a 95% uncertainty interval by resampling entire tropical cyclones with replacement^[3]. Whenever a storm is resampled, all associated storm-time pairs are added to the sample. This resampling procedure is conform the statistical structure of the data and approximates the error w.r.t. taking the average over infinitely many storms. If the 95% uncertainty interval for a square includes zero, “ $\approx$ ” is depicted in the middle of the square.

I.4 Comparison against operational IFS HRES at 0.1°

We include roll-out plots of Aurora at 0.1° compared to HRES forecasts at 0.1° for headline variables (Figure I29) and the lower atmosphere (Figure I30). Due to the lack of 0.1° forecasts in the upper atmosphere, we include as a preliminary result a scorecard versus HRES 0.25° in Figure I31. Although this is not an apples-to-apples comparison, HRES 0.25° RMSEs tend to be lower than HRES 0.1° RMSEs since higher-resolution models have more fine-grained details that they need to get right. Therefore, the scorecard provides a good approximation to the performance of Aurora 0.1° in the upper atmosphere.

Aurora 0.1° model trained with no pretraining. To demonstrate the impact of pretraining we also include a scorecard (Figure I32) comparing the performance of Aurora 0.1° fine-tuned versus the model trained from scratch on the 0.1° data. The fine-tuned model consistently outperforms a model trained from scratch despite considerable effort being expended to optimise model training without first pretraining.

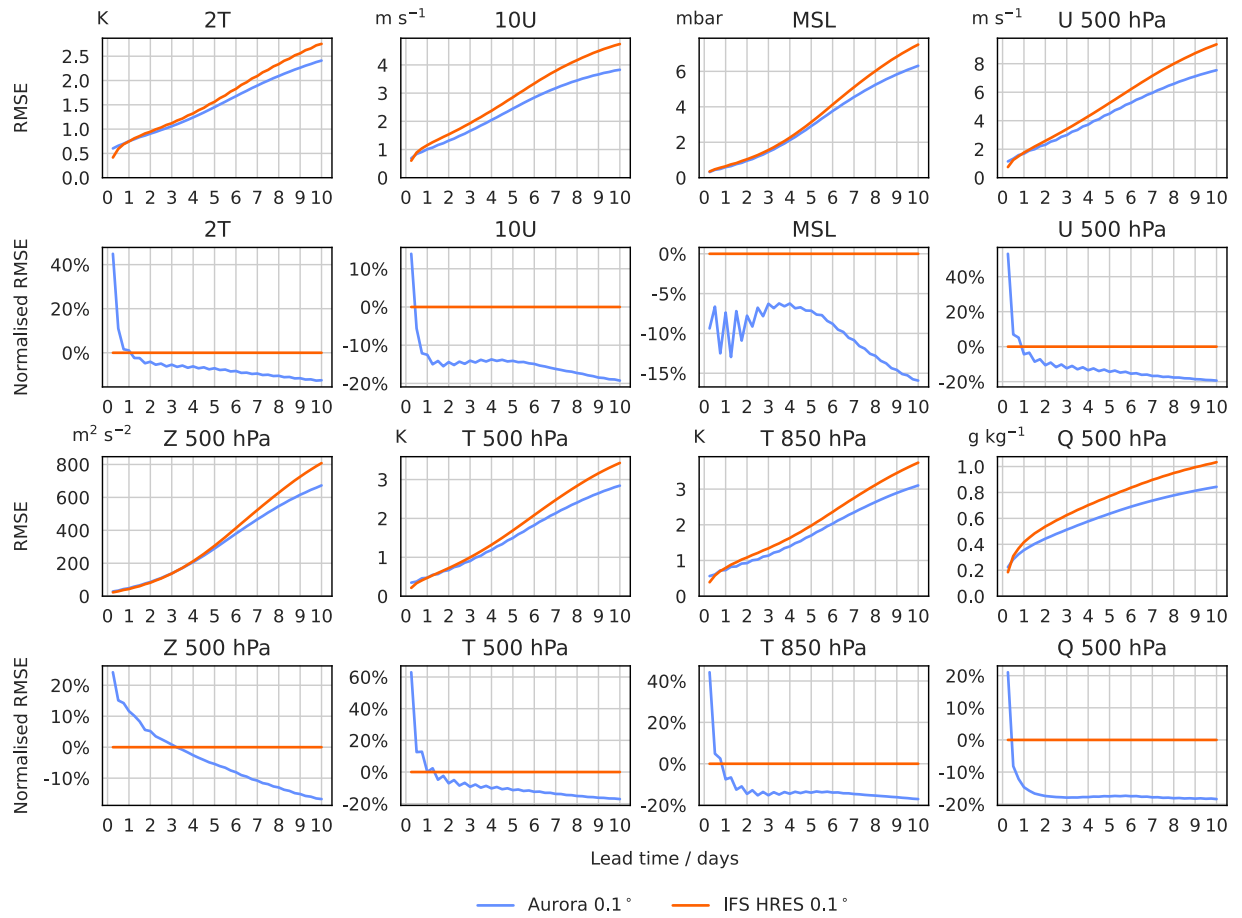


Fig. I29: Root mean square error (RMSE) of Aurora 0.1° compared to HRES 0.1° for the headline variables. Shows both unnormalised and normalised RMSEs.

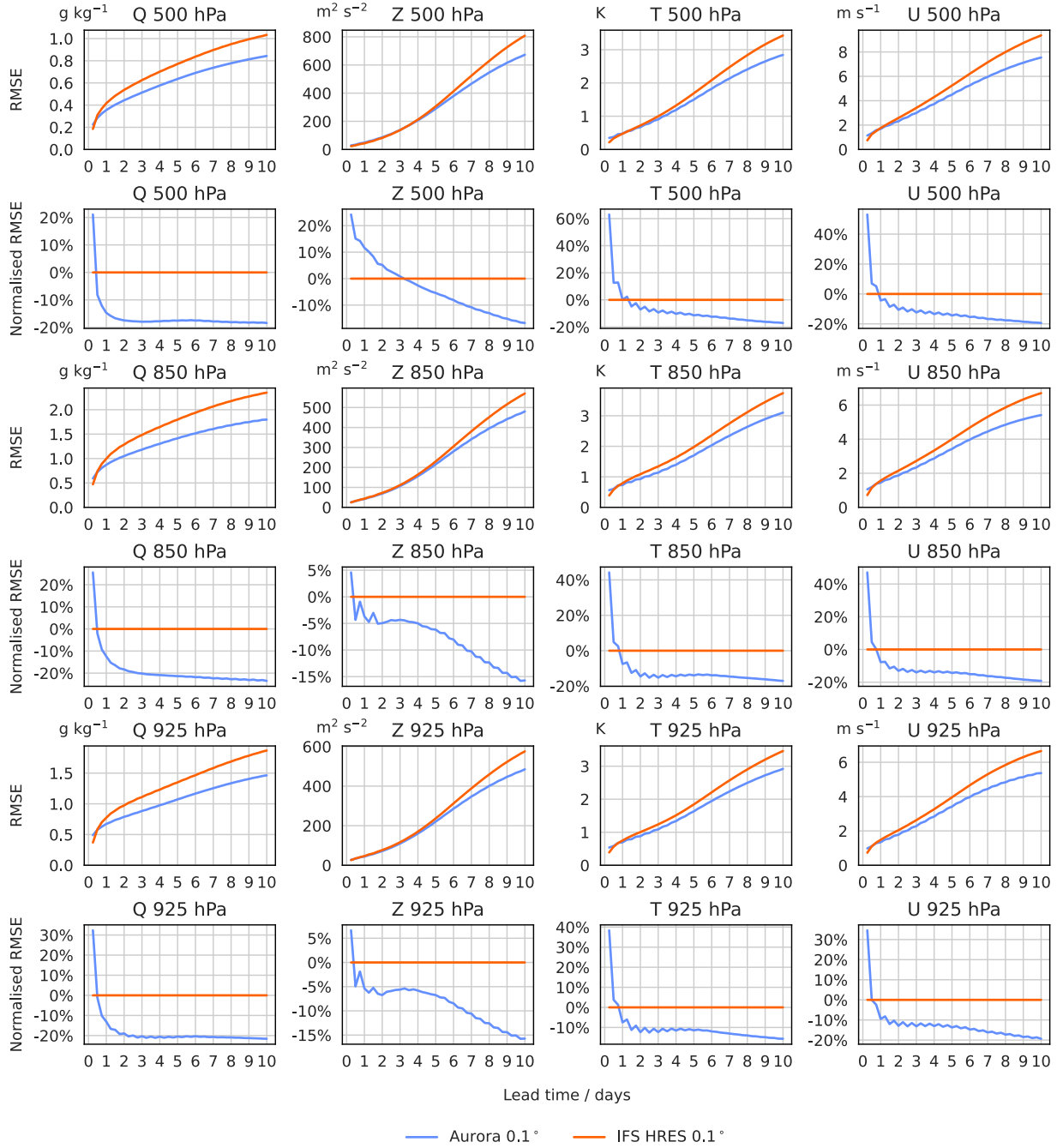


Fig. I30: Root mean square error (RMSE) of Aurora 0.1° compared to HRES 0.1° in the lower atmosphere. Shows both unnormalised and normalised RMSEs.

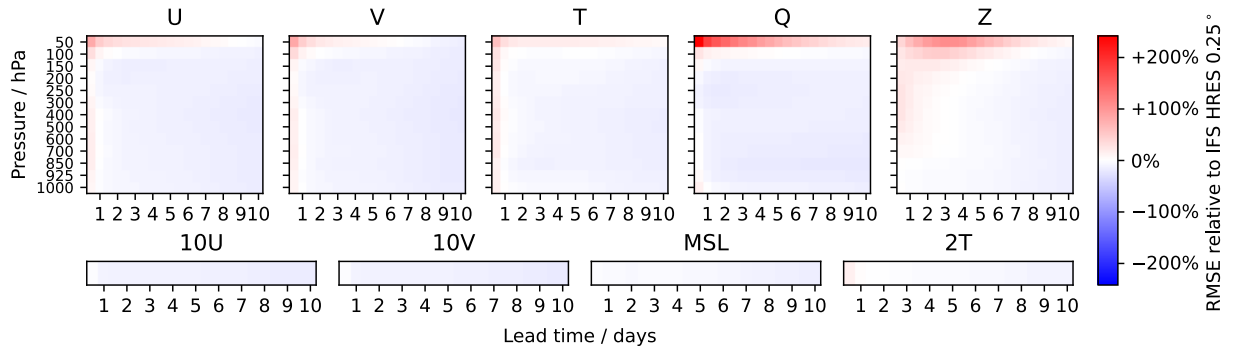


Fig. I31: Scorecard comparing Aurora 0.1° versus HRES 0.25° across all levels. We note that the RMSEs of 0.25° HRES are generally lower than those of 0.1° HRES. Nonetheless, Aurora shows good performance and generally outperforms HRES also in the higher atmosphere.

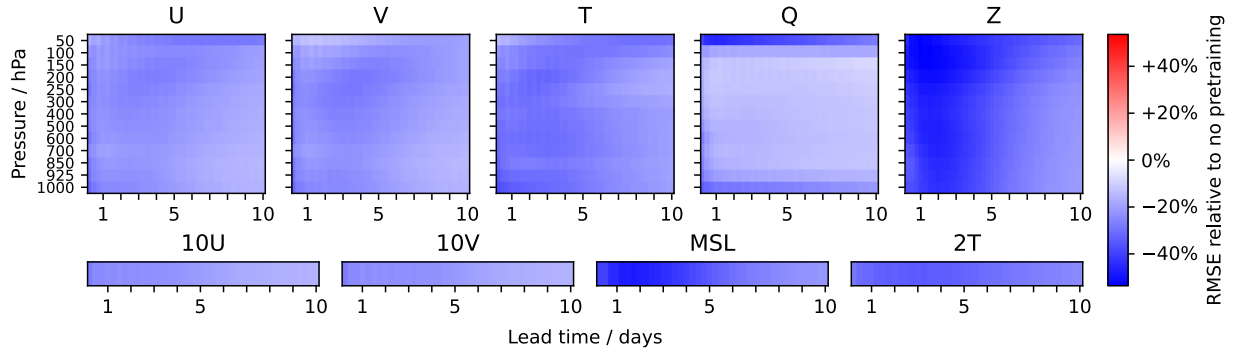


Fig. I32: Scorecard comparing Aurora 0.1° versus Aurora 0.1° trained from scratch with no pretraining. Aurora 0.1° consistently outperforms the model with no pretraining. On average, pretraining improves performance by 25%.

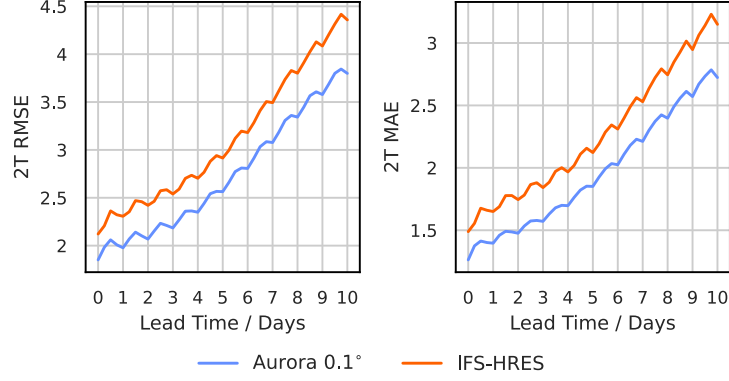


Fig. I33: 2 m temperature performance at weather station observation points for Aurora 0.1° versus IFS HRES 0.1° (RMSE and MAE). Due to the different initialisations, Aurora has an advantage. Nonetheless, Aurora’s forecast performance degrades at roughly the same rate as HRES as the lead time increases.

I.5 Comparison against weather station measurements

Most verification efforts for AIWP models have focused on gridded reanalysis data such as ERA5^[60]. Not only are weather station observations more reflective of weather experienced by people, but they can also reveal deficiencies in models relying on diagnostic parameters generated by the NWP models producing the analysis. Therefore, we also evaluate Aurora’s forecasts on raw observations from the WeatherReal-ISD dataset^[46]. This evaluation includes daily forecasts issued at 00 UTC between 2 January 2023 and 19 December 2023. Aurora outperforms IFS HRES for 10 m wind speed forecasts across all lead times up to 10 days (Figure 5b). Further results for 2m temperature are available in Supplementary F.2. This result shows how a global high-resolution AIWP model can set a new state-of-the-art as measured by real weather observations.

Aurora is initialised from the official analysis product, while forecasts are initialised from HRES T0. The former includes an additional assimilation step for the surface, which improves the 2T estimate. Therefore, Aurora has an unfair advantage in terms of initial conditions when compared on 2T (Figure I33). Nonetheless, the errors of HRES and Aurora remain parallel as the lead time increases which shows that the forecast quality of Aurora does not degrade over time compared to HRES.

I.6 Benefits of modelling at 0.1° resolution

In Section 6, we fine-tune Aurora for high-resolution predictions at 0.1°. Instead of fine-tuning at 0.1° resolution, one could ask why not simply fine-tune at 0.25° and simply interpolate the predictions to 0.1° resolution? The benefit of modelling at 0.1° resolution is that the higher resolution can better represent physical processes at small scales, and hence predictions at 0.1° can capture these effects, whereas predictions at 0.25° could struggle. The meso- β (20 km to 200 km) scale exactly represents physical processes that 0.1° should be able to capture but 0.25° might struggle with, such as mesoscale convective systems and complexes or mountain and valley breezes. In this appendix, we provide numerical evidence that fine-tuning at 0.1° produces superior predictability for physical processes at the meso- β scale.

Methodology. We compare the predictability of Aurora 0.1° to Aurora 0.25° regridded to 0.1° using the PyPI package `xesmf` on three different meteorological scales: the meso- β scale (20 km to 200 km), the meso- α scale (200 km to 2,000 km), and the synoptic scale (2,000 km and larger). Predictability is measured with the coefficient of determination, R^2 , which quantifies precisely which proportion of the data explained by the predictions. We take HRES analysis at 0.1° as the ground truth. To disentangle R^2 at different meteorological scales, we use the spherical harmonic transform from `torch-harmonics`^[44] to decompose the prediction error for both models into errors at different wavelengths. We then select the appropriate wavelengths for every meteorological scale, and sum the squared amplitudes back to a total MSE for that scale. The MSE is divided by the total variance of the ground truth in the same wavelength band, and the resulting ratio is equal to $1 - R^2$. The difference in R^2 , $R^2_{\text{Aurora } 0.1^\circ} - R^2_{\text{Aurora } 0.25^\circ}$, finally quantifies the difference in predictability between Aurora 0.1° and Aurora 0.25° regridded to 0.1°.

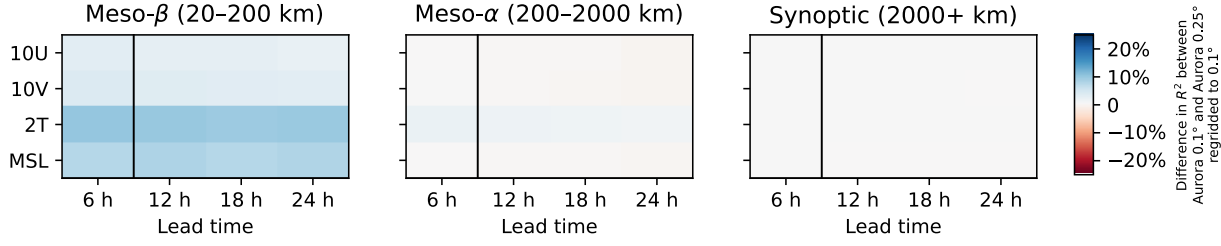


Fig. I34: At various meteorological scales, comparison of R^2 , a measure of predictability, of Aurora 0.1° to Aurora 0.25° regrided to 0.1° resolution. IFS HRES analysis at 0.1° is used as the ground truth. R^2 is estimated from predictions for UTC 00 and UTC 12 from the period Jan 2023 to Mar 2023 inclusive. Blue means that Aurora 0.1° has better predictability, and red means that Aurora 0.25° regrided to 0.1° has better predictability.

Results and conclusion. Aurora 0.1° demonstrates substantially improved predictability for meso- β -scale physical processes (a 5.4% increase in R^2 on average), whereas predictability for meso- α -scale and synoptic physical processes remain roughly unchanged (differences of 0.3% and 0.03% on average) (see Figure I34). That Aurora 0.1° shows substantially improved predictability on the meso- β scale is not a surprising finding, because this is exactly the scale that a resolution of 0.25° cannot effectively capture, but a resolution of 0.1° can.

I.7 Storm Ciarán

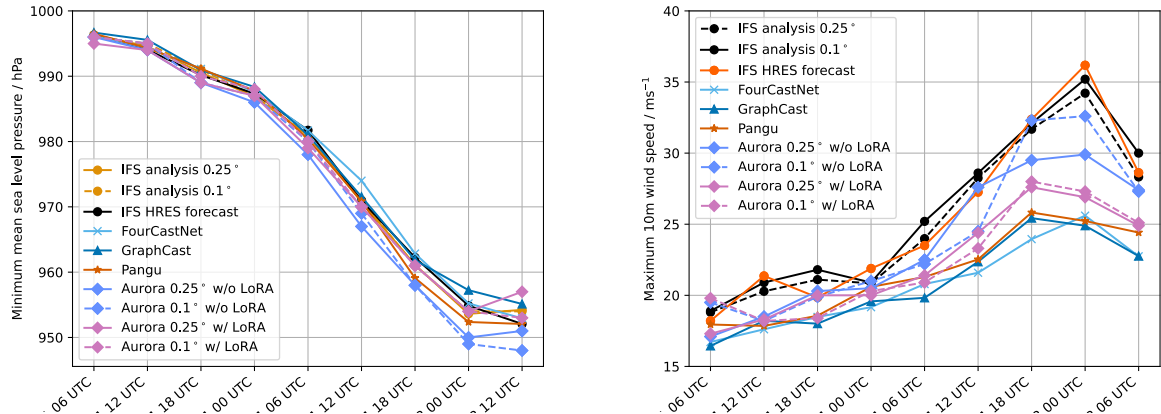


Fig. I35: Minimum mean sea level pressure and maximum 10 m wind speed for Storm Ciarán, at 0.25° and 0.1° resolution, both with and without LoRA fine-tuning. All models perform similarly at predicting sea level pressure (left). In contrast, Aurora with and without LoRA outperform the other AI models in capturing the higher amplitude of the wind speeds during the storm (right). Nevertheless, LoRA has a smoothing effect which results in a lower wind speed forecast.

Table I7: Bounding boxes that track Storm Ciarán. We used these bounding boxes to compute the maximum 10 m wind speed and minimum MSL.

Time range	Latitudes	Longitudes
2023-10-30 18 UTC to 2023-10-31 06 UTC	[35°, 48°]	[−75°, −48°]
2023-10-31 12 UTC to 2023-11-01 00 UTC	[40°, 48°]	[−50°, −25°]
2023-11-01 06 UTC to 2023-11-01 12 UTC	[35°, 50°]	[−65°, −10°]
2023-11-01 18 UTC to 2023-11-02 00 UTC	[35°, 60°]	[−25°, 10°]
2023-11-02 06 UTC and later	[35°, 60°]	[−10°, 10°]

Storm Ciarán was a high-impact windstorm which took place across North-West Europe in late 2023. In Figure 5d, we reproduce the findings by^[48] and include predictions from Aurora at 0.1° resolution. All models are initialized on 31 Oct 00 UTC using the latest cycle of IFS analysis, CY48R1. We observe that among the AI models tested, Aurora stands out as the only one capable of accurately predicting the abrupt rise in maximum 10 m wind speed. It forecasts approximately 7 ms^{-1} higher than the other AI techniques, closely matching IFS analysis, which we take to be the ground truth. Notably, FourCastNet^[38], GraphCast^[3], and Pangu-Weather^[2] are constrained to a 0.25° resolution, potentially hindering their ability to capture rapid onset phenomena. However, even at 0.25° resolution, Aurora demonstrates some capability in capturing this sharp increase, albeit to a lesser degree than its 0.1° counterpart; see Figure I35.

Figure 5c shows Storm Ciarán’s intensification over a 24 hour period, where we compare Aurora forecasts (top) to IFS analysis (bottom) visually in terms of 10 m wind speed. We also plot the minimum MSL, corresponding to the center of the storm, and find that the Aurora’s forecasts are similar to IFS HRES analysis. The primary difference lies in the level of detail: IFS analysis captures higher spatial frequency content, whereas Aurora’s forecasts are smoother, although less so than those from the 0.25° model. Despite this disparity, the forecasts are visually similar.

Two versions of Aurora. We compare Aurora variants with and without LoRA fine-tuning for predicting Storm Ciarán. Figure I35 shows minimum mean sea level pressure (left) and maximum 10 m wind speed (right), including the two variants of Aurora, namely, with and without LoRA fine-tuning, at both 0.25° and 0.1° resolution. We evaluate all methods based on how closely they resemble IFS analysis at the corresponding resolution, and find that, as expected, the Aurora variants at 0.1° resolution clearly outperform those at 0.25° resolution.

For minimum mean sea level pressure, the variants with LoRA fine-tuning perform better (i.e., align more closely with IFS analysis). However, for maximum 10 m wind speed, the variants without LoRA fine-tuning perform better, capturing the peak wind speeds more accurately. We see that Aurora at 0.1° (with LoRA fine-tuning) aligns almost perfectly with IFS analysis at 0.1° . In Section 6, we showed that Aurora is the only AIWP model capable of capturing the peak in maximum 10 m wind speed on 2 November 2023 at 00 UTC, where the best version of Aurora is at 0.1° , without LoRA fine-tuning. Table I7 tabulates the bounding boxes used to compute the maximum 10 m wind speed and minimum MSL.

We postulate that LoRA is useful when computing statistics of smooth quantities with less extreme activity (e.g., MSL), where the loss of detail and additional smoothing is less critical; while turning LoRA off is better for computing statistics such as wind speed, where the loss of detail can substantially affect the statistics by suppressing the extrema. This behavior is further supported by our spectral analysis in Supplementary I.8, where we show how LoRA affects the power spectrum of the model’s predictions at different lead times (Figure I38).

In summary, the use of LoRA during roll-out fine-tuning provides us with a mechanism to control the trade-off between RMSE performance and the retention of high-frequency details in long-term predictions. This flexibility is valuable in operational settings where different types of predictions may be required for different applications.

I.8 Power spectra

To assess the quality of predictions, a useful characteristic is the power spectrum. This section contains plots of power spectra for predictions and targets calculated using the method described in^[3].

At short wavelengths (i.e. high frequencies) Aurora’s forecasts are uniformly higher in power than those of GraphCast, which means less blurring (Figure I36). In the mid-range of the spectrum, Aurora’s forecasts have higher power than those of GraphCast for lead times of 1 and 2 days, but then drop in power below GraphCast’s forecasts with 5 and 10 days lead times. In general, as more LoRA steps are added, the power in the forecast is reduced, especially in the mid-range of the spectrum (Figure I38). For the longest lead time of 10 days and with 40 LoRA steps, there is a clear reduction in power at the short wavelengths (i.e. high frequencies) which results in blurring.

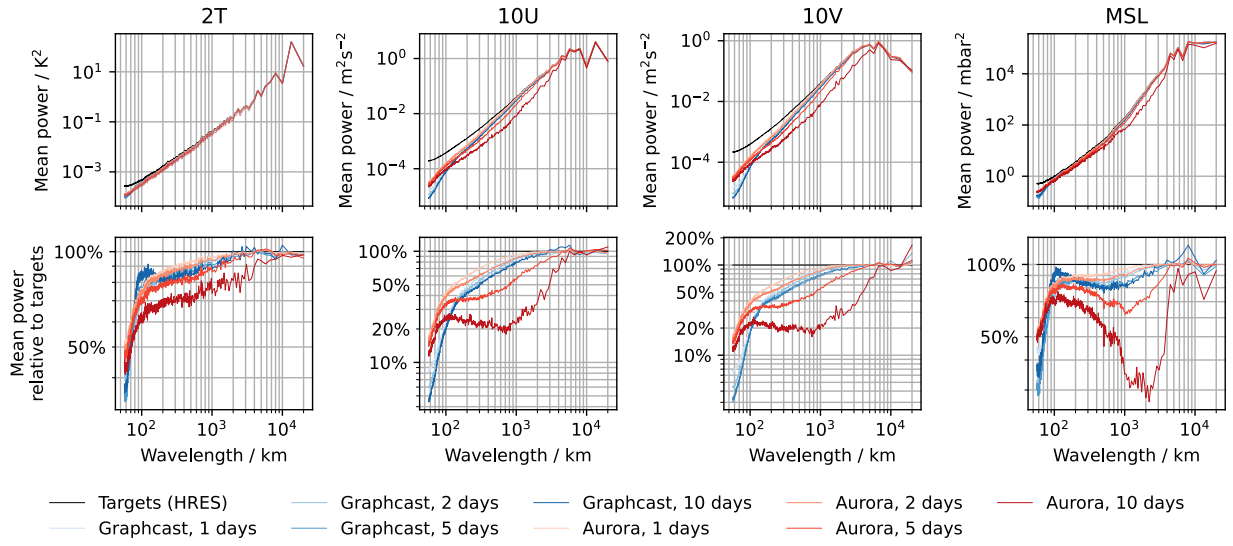


Fig. I36: Power spectra of Aurora's predictions and GraphCast's predictions for 2022. The first row plots the mean power for predictions. The columns represent 2T, 10U, 10V, and MSL, respectively. The second row plots the mean power relative to the HRES targets. Each individual chart illustrates the change in the power spectrum for lead times of 1, 2, 5, and 10 days.

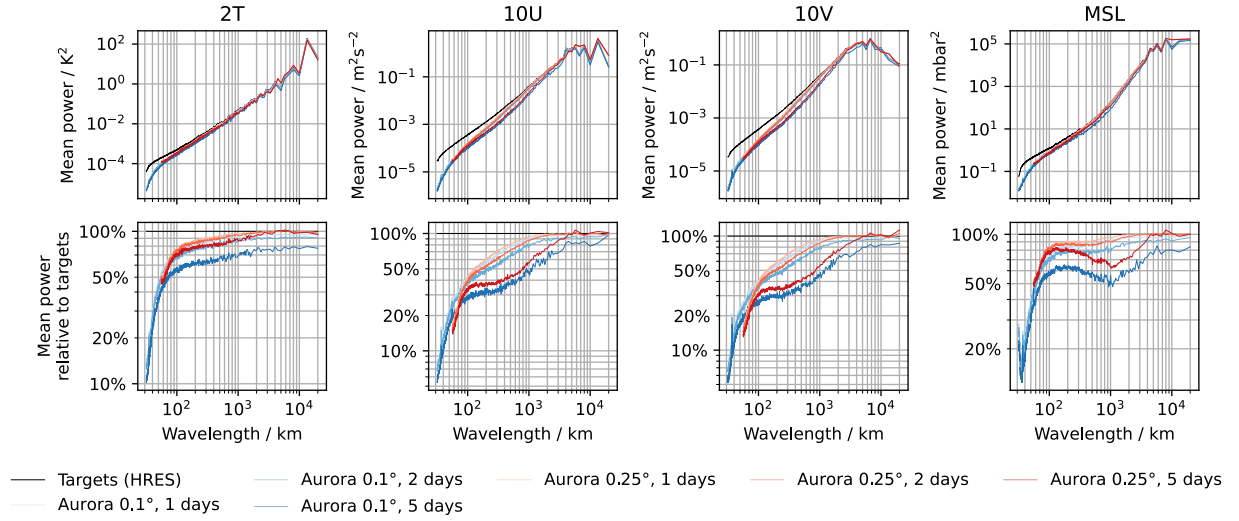


Fig. I37: Power spectra of Aurora's predictions at both 0.1° and 0.25° resolutions for January 2022. The first row plots the mean power for predictions. The columns represent 2T, 10U, 10V, and MSL, respectively. The second row plots the mean power relative to the HRES targets.

When comparing the spectrum of Aurora between 0.25° and 0.1° , the power of the 0.1° forecasts is generally lower than the power of the 0.25° forecasts. However, the power of the 0.1° forecasts extends to shorter wavelengths (i.e. higher frequencies) than 0.25° forecasts, reflecting the higher resolution of the latter. These observations are in agreement with the discussion provided in Supplementary I.7, where we pointed out that the difference between Aurora forecasts with and without LoRA fine-tuning can be viewed analogously to the difference between the forecasts produced by the IFS ensemble mean and IFS HRES.

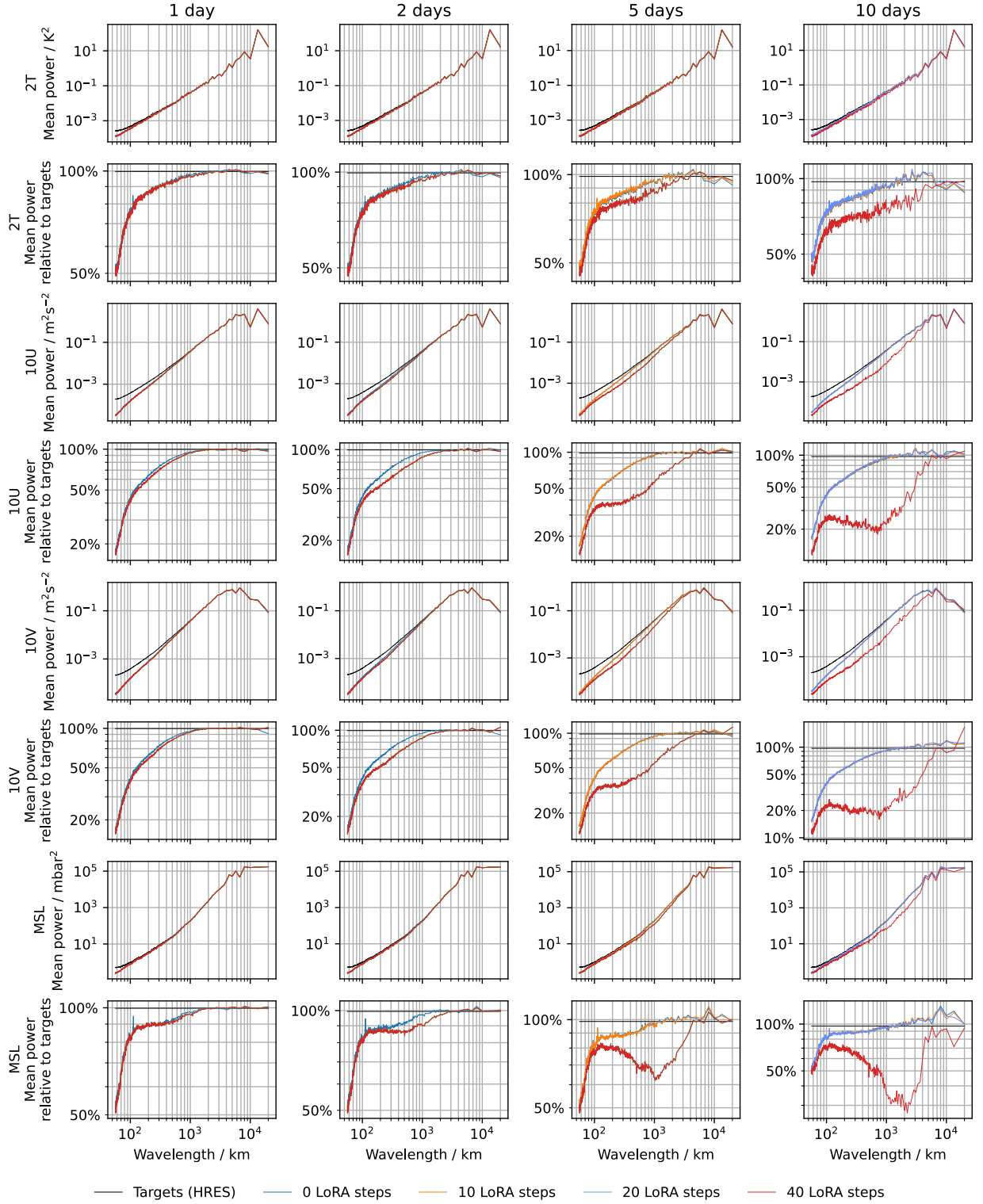


Fig. I38: Power spectra of Aurora's predictions with different numbers of LoRA steps for 2022. The first row plots the mean power for 2T predictions. Each column represents a different lead time of 1, 2, 5, or 10 days, respectively. The second row plots the mean power for 2T relative to the HRES targets. Similarly, subsequent pairs of plots represent, respectively, the power and relative power of predictions of 10U, 10V, and MSL.

