Weather Forecast Models Characteristics

林志偉 B11611017

Weather forecast models:

- Google DeepMind's GraphCast
- Huawei's Pangu-weather
- NVIDIA's FourCastNet

NWP models vs ML Weather models:

Numerical Weather Prediction models use Complex Equations (physical principles, fluid dynamics and thermodynamics, such as the Navier-Stokes equations)

NWP models start with a snapshot of the current state of the atmosphere (initial conditions), gathered from observational data (satellites, weather stations, etc.), and then solve these physical equations over time to predict future weather.

For vertical levels, discretize the atmosphere into many vertical levels, often over 100, from the surface to the upper atmosphere (~137 levels for ECMWF), to capture the vertical profile of weather systems.

Solving these equations requires massive computational power and can take hours on supercomputers. The more detailed the resolution, the more intensive the computation.

Machine Learning (ML) Weather Models:

ML models do not use explicit physical laws (e.g., Navier-Stokes). Instead, they approximate the underlying processes through patterns found in the data

ML models learn patterns from historical weather data, satellite imagery, sensor data, and other sources. Instead of solving physics equations, they identify statistical relationships in the data to predict future weather.

Supervised Learning: Models are trained on labeled datasets (e.g., historical weather inputs and their corresponding outcomes).

Once trained, ML models can produce forecasts much faster than traditional models, potentially in real time, as they bypass the need for solving complex physical equations.

Pangu Weather Forecast Model

Pangu-Weather14 consists of four deep neural networks with different lead times (time between input and output) of 1 h, 3 h, 6 h and 24 h.

5 upper atmosphere and 4 surface variables at 13 different pressure levels were used to train the model with a combined total of 256 million parameters.

Main Methods:

- Deep Learning Techniques:
- Self-attention mechanism: The core of Pangu-Weather's architecture is the Transformer model, which uses self-attention to capture dependencies between different parts of the input. Unlike conventional recurrent models or convolutional models, Transformers can capture long-range dependencies more efficiently. In weather forecasting, these dependencies include the relationships between different regions of the atmosphere and how they evolve over time.
- The overall deep network architecture is called 3DEST or 3D-Earth-specific Transformer that integrates height information into a new dimension thus capturing relationships between atmospheric variables across pressure levels
- Data is fed into the neural network and a process called patch embedding is used to downsample the input data from individual grid cells into a 3D cube.
- This cube is then put through an encoder-decoder based on a ViT called the Swin transformer25 with 16 blocks.
- The positional bias in the Swin transformer is replaced with an Earth-specific positional bias to reflect the fact that in a 2D projection of a sphere, distances between neighbouring points are not the same across all latitudes.
- Hierarchical resolution: The model predicts at different spatial resolutions. It can first generate a coarse prediction (e.g., for large-scale atmospheric movements like jet streams) and then refine these predictions at a finer scale to capture localized events (e.g., storms, rainfall). This helps it to provide highly accurate predictions at both macro and micro levels.

2. NVIDIA Weather Forecast Model

FourCastNet, short for Fourier ForeCasting Neural Network

accurate short to medium range global predictions at 0.25◦ resolution.

(0.25° resolution means the model divides the globe into small boxes, with each box having a size of 0.25 degrees in latitude and 0.25 degrees in longitude.)

generates a week long forecast in less than 2 seconds

1. Basic Strategy: Recursive Forecasting

- FourCastNet makes forecasts by recursively stepping forward in time. It uses a deep learning model that is fed with data representing the current state of the atmosphere, and the model predicts future states over time in steps.
- FourCastNet recursively applies an Adaptive Fourier Neural Operator (AFNO) network to predict their dynamics at later time steps.

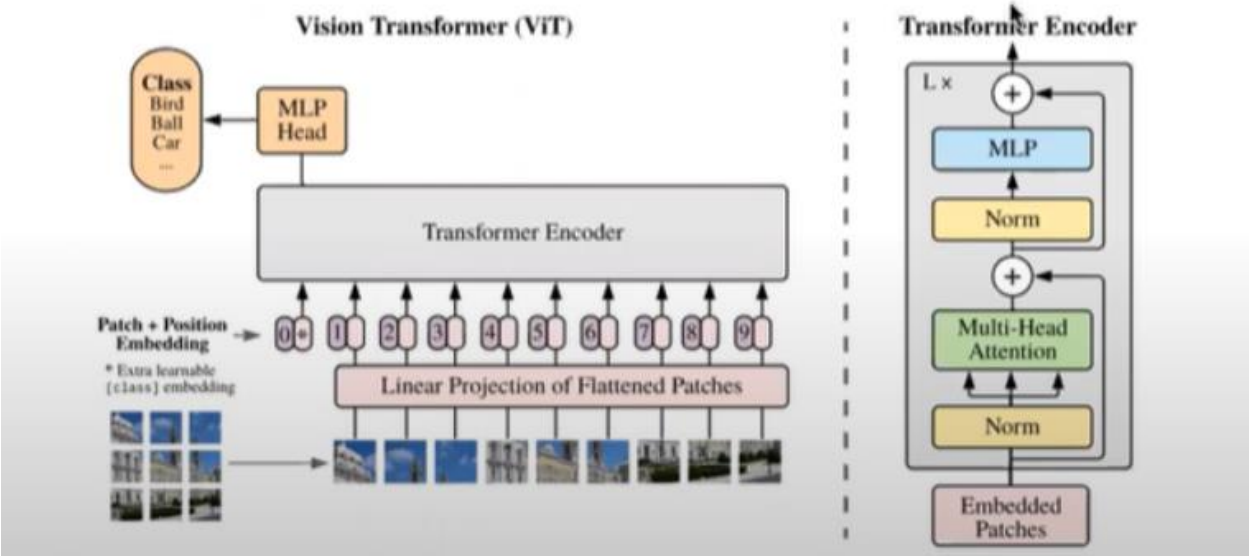
2. Characterizing the Atmospheric State

- The model relies on a few key variables that characterize the atmospheric state, such as wind speed, temperature, pressure levels, etc.
- Initially, the model uses a subset of variables (those listed in the table), but in the future, it could potentially include all variables used by Numerical Weather Prediction (NWP) models.
- These variables may provide enough information for the model, without needing a full NWP-like characterization of the state.

3. Training and Inference

- Training: The model learns to predict the target state X(t + dt) from the input X(t), where dt is a time increment (e.g., one hour, six hours, etc.).
- In the second fine-tuning or inference step, the pre-trained model is used to produce inferences from a defined state $\mathbf{X}(t)$, first for $\mathbf{X}(t + \Delta t)$ and this output from the models is itself then used to generate $\mathbf{X}(t + 2^* \Delta t)$ or the output for the second time step.
- Inference: During actual forecasting, the model performs an autoregressive forward step. This means that it predicts the state at a future time, then uses that predicted state as the input to predict the next step, recursively forecasting further into the future.

$$C \times H \times W \rightarrow N \times P^2C \rightarrow N \times D$$



FourCastNet is based on a vision transformer backbone

Typically used for image recognition tasks.

Vision Transformers are deep learning models originally developed for image recognition tasks. They treat images as sequences of patches, similar to how Transformers handle words in natural language processing.

In FourCastNet, the input (global atmospheric variables) is **tokenized** by splitting it into **patches** with positional embeddings, similar to how images are processed in ViTs.

The atmospheric input (e.g., wind, temperature, pressure data across a global grid) is divided into smaller patches of data, each representing a section of the Earth's surface with multiple variables.

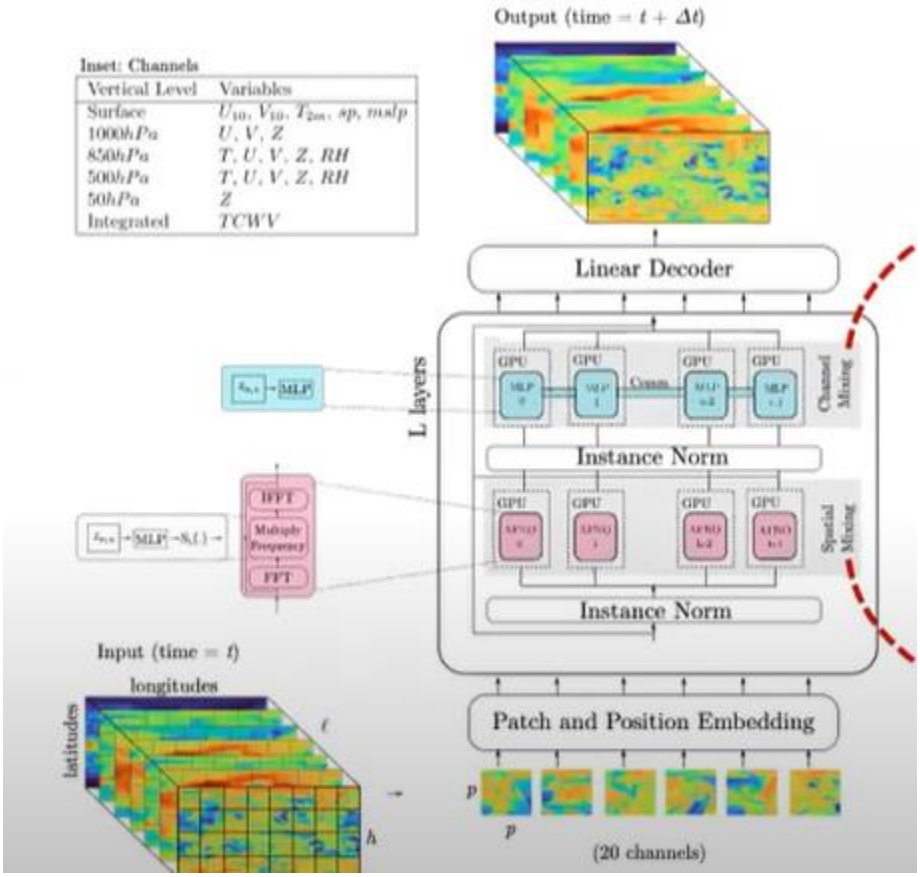
The input tensor dimensions follow the pattern:

- **C × H × W** refers to the number of channels (C, the different atmospheric variables), and the height and width (H and W) of the grid or image.
- The transformer reshapes this into **N × P^2C**, where **N** is the number of patches, and each patch has **P^2C** elements (with **P^2** being the size of the patch).
- Finally, this is transformed into **N × D**, where **D** is the **embedding dimension** (a fixed size vector for each patch)

After patch embedding, the **Transformer Encoder** processes these tokenized patches:

- **Multi-Head Attention** layers allow the model to capture relationships between different patches, i.e., how different regions of the atmosphere interact or influence each other over space.
- **MLP (Multi-Layer Perceptron)** layers further process these attention outputs.
- **Normalization** layers (Norm) help stabilize and standardize the model's internal representations during training.

The transformer's computational complexity scales **quadratically with the input size**. This means that as the number of patches increases, the amount of computation grows quadratically.



Feature Parallelism

Spatial Parallelism

What is split between the GPUs:

Feature Parallelism: Feature or channel dimension

Spatial Parallelism: Spatial dimensions (Height, Width, etc.)

Feature Parallelism:

- **Channel mixing MLPs (Multi-Layer Perceptrons)**:
    - The model uses **MLPs** to handle "channel mixing," which refers to processing different features (variables like wind, temperature, etc.) in parallel across GPUs.
    - These MLPs split the **hidden dimension** (the internal representation of the data) across multiple GPUs, enabling the workload to be distributed.
    - However, this requires **communication between GPUs** during both forward and backward passes, meaning that the GPUs need to exchange information while processing to maintain synchronization.

Spatial Parallelism in FourCastNet:

- **FFT and Large Grids**:
    - FourCastNet leverages the **Fast Fourier Transform (FFT)** to capture spatial dependencies in weather data. FFTs are computationally expensive operations, especially on high-dimensional grids.
    - Spatial parallelism is employed to split the **height (H)** and **width (W)** dimensions of the data grid across multiple GPUs. For instance, one GPU might handle part of the grid representing a certain region of the globe, while another GPU handles a different region.
- **Distributed FFTs**:
    - Once the grid is split, each GPU computes FFTs on its portion of the grid. This means the Fourier transforms (which are used to model long-range dependencies across the globe) can be computed in parallel on smaller subregions.
    - After the local FFTs are computed, GPUs need to exchange data because FFTs inherently require information from the entire spatial domain (i.e., weather data at different locations are interdependent). This is where **communication between GPUs** (e.g., using **all-to-all communication** operations) comes in.

- **Layer Norms and Synchronization**:
    - During training, certain operations like **layer normalization** or **gradient updates** need to take into account the entire spatial domain (i.e., the complete weather grid).
    - After GPUs process their portions of the data, they must exchange statistics such as the mean and variance of the data (for normalization purposes), which requires careful synchronization between the distributed processors.

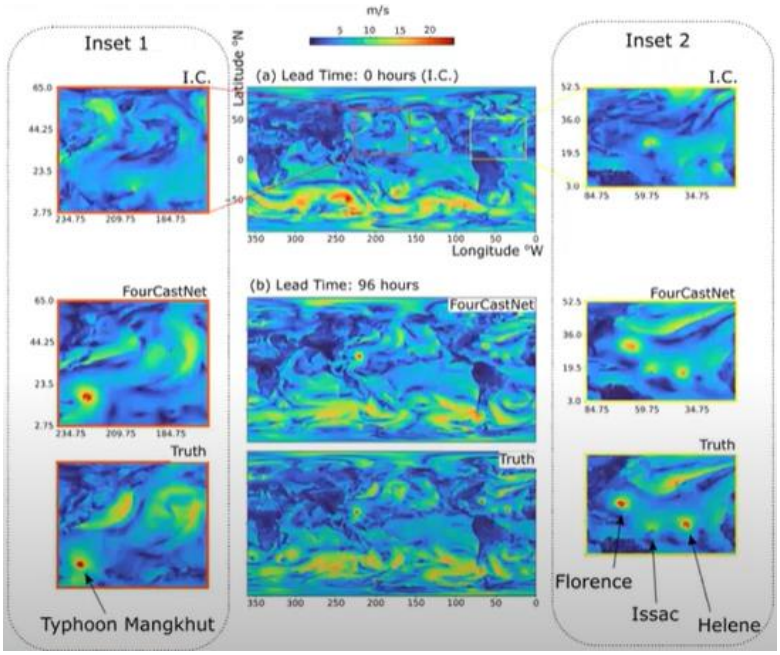
- **Why Spatial Parallelism?**
    - Spatial parallelism is crucial for FourCastNet because of the sheer size of weather data grids. Handling all spatial points on a single GPU would not only be computationally infeasible but would also exceed memory limits.
    - By splitting the grid spatially, the computation becomes more manageable, allowing FourCastNet to process high-resolution weather simulations efficiently and scale across many GPUs.

Table 4 *FourCastNet modeled variables*

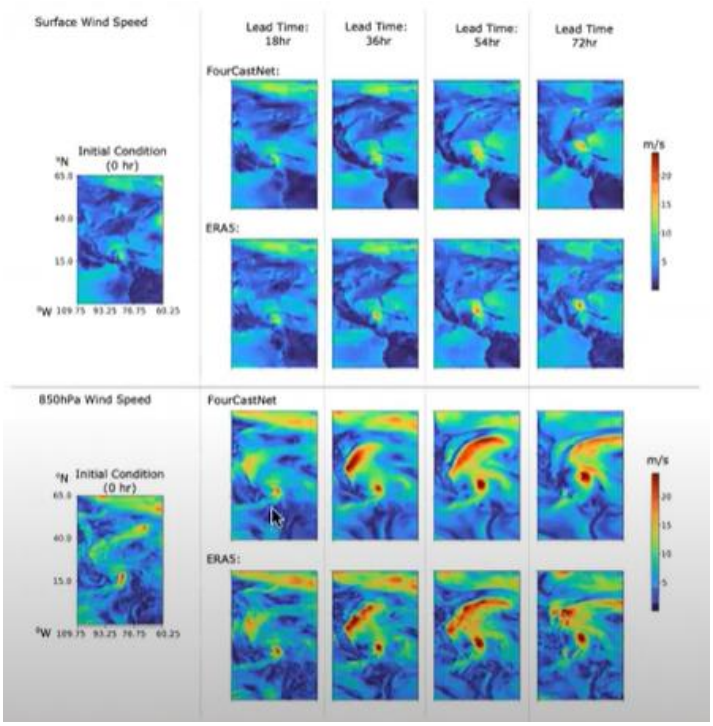
Vertical Level	Variable
Surface	U10, V10, T2M, SP, MSLP
1000 hPa	U, V, Z
850 hPa	T, U, V, Z, RH
500 hPa	T, U, V, Z, RH
50 hPa	Z
Integrated	TCWV

Results:

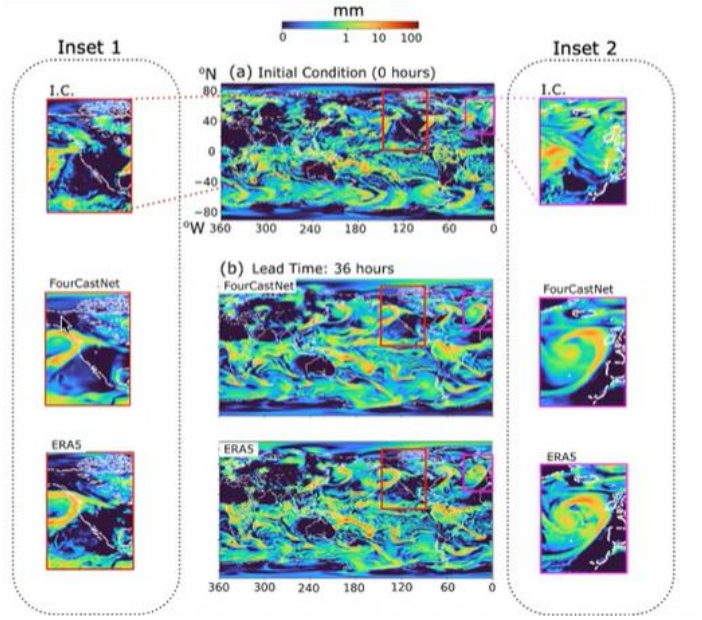
- Excellent skill in predicting surface wind velocities



- Can predict hurricane paths and intensities



- Comparable skill to IFS in total precipitation prediction, better at short time intervals



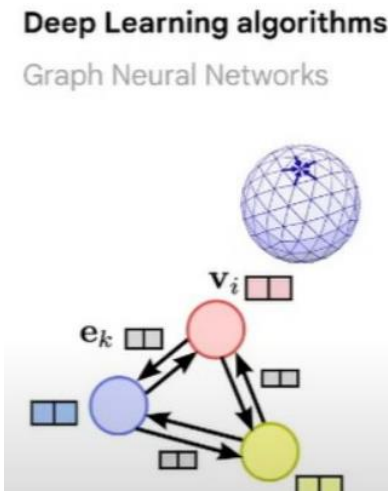
Fourcastnet is faster and more efficient than NWP

Latency and Energy consumption for a 24-hour 100-member ensemble forecast				
	IFS	FCN - 30km (actual)	FCN - 18km (extrapolated)	IFS / FCN(18km) Ratio
Nodes required	3060	1	2	**1530**
Latency (Node-seconds)	984000	7	22	**44727**
Energy Consumed (kJ)	271000	7	22	**12318**

3. GraphCast Weather Forecast Model

Main Methods:

- GraphCast takes as input the two most recent states of Earth's weather—the current time and six hours earlier—and predicts the next state of the weather six hours ahead.
- The model is trained with 5 surface and 6 atmospheric variables at 37 pressure levels resulting in 227 variables for every data point or grid cell.



- **Graph Neural Networks (GNNs)**:
    - Makes the Earth a mesh, nodes have latent vector on each node and edge of the graph, nodes inform their neighbours and update their latent vector by sending messages to nearby neighbours.
    - Benefits: Locality, in short period of time weather in Taipei will only depend on weather in Taiwan.
    - Equivariance, the weather function used in Brazil would be the same on used in Taipei, more parameter efficient method.
- Extreme weather: tropical cyclones

  Reimplemented the cyclone tracker used in EMCWF
- Extreme weather: atmospheric rivers

  One day of accuracy over HRES
- Extreme weather: extreme heat

HRES better than graphcast at shorter lead times, GraphCast better at longer lead times.

- **Data-Driven Approach**:
    - Emphasizes learning from extensive observational data instead of solely relying on physical models, leading to improved performance in dynamic conditions.
    - Leverages vast datasets from diverse sources to enhance model robustness.

ECMWF variables used in datasets:

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

Required Input Fields:

- ERA5 Reanalysis: Dataset of weather information from 40+ years organized by the ECMWF.
- Spatial data: grid-based representation of the Earth's surface, including land use and vegetation cover.
- Climate model outputs: used to initialize the graph structure and inform interactions between nodes.
- Real-time meteorological data: updates for ongoing predictions to refine and adjust forecasts dynamically

Similarities and Differences

Feature	Pangu	FourCastNet	GraphCast
Main Methods	Deep learning (CNNs, RNNs), data assimilation	GPU acceleration, physics-based modeling, ensemble forecasting	Graph neural networks (GNNs), data-driven approach
Computational Focus	Emphasis on deep learning techniques	High emphasis on GPU performance and NWP	Focus on graph-based representation and learning
Computational cost	High	Low	Medium
Forecasting ability	3-10 days	Up to 10days	Up to 10 days
Transformers	Yes	Yes	No
Forecasting speed	**200x faster** than traditional NWP models	**1,000 to 45,000 times faster** than traditional NWP models	Slower than Fourcastnet and Pangu, faster than NWP models

Model	Training Cost	Inference Cost
FourCastNet	**Lower training cost** due to FNO efficiency. FFT reduces spatial complexity	**Very low inference cost**: Global weather forecasts in seconds using GPUs
Pangu-Weather	**High training cost**: Transformer-based, requiring large compute resources and long training times	**Moderate inference cost**: Transformer layers add complexity, but still much faster than traditional models
GraphCast	**Moderate to high training cost**: Graph-based computations require complex memory access and higher compute power	**Moderate inference cost**: Slower than FNOs but faster than traditional models