Welcome

Discover Anemoi

A series of training webinars

Webinar 2: Anemoi Graphs + Models

Wednesday, 17 June 2026

Welcome

Discover Anemoi

A series of training webinars

Webinar 1: Introduction + Anemoi Datasets – 16 June 2026 at 15 CEST

Webinar 2: Anemoi Graphs + Anemoi Models - 17 June 2026

Webinar 3: Anemoi Training + Anemoi Inference - 18 June 2026 at 15 CEST

Anemoi Graphs + Anemoi Models

Presentations - by *Mario Santa Cruz*

Questions & Answers – dedicated session after each presentation webinar, via the chat

Recordings – this webinar is being recorded and will be made available on the event webpage

No AI meeting assistants – please don't add them for this webinar

Contact – training@ecmwf.int

DISCOVER ANEMOI · WEBINAR SERIES · PART 2

Anemoi graphs

PRESENTER

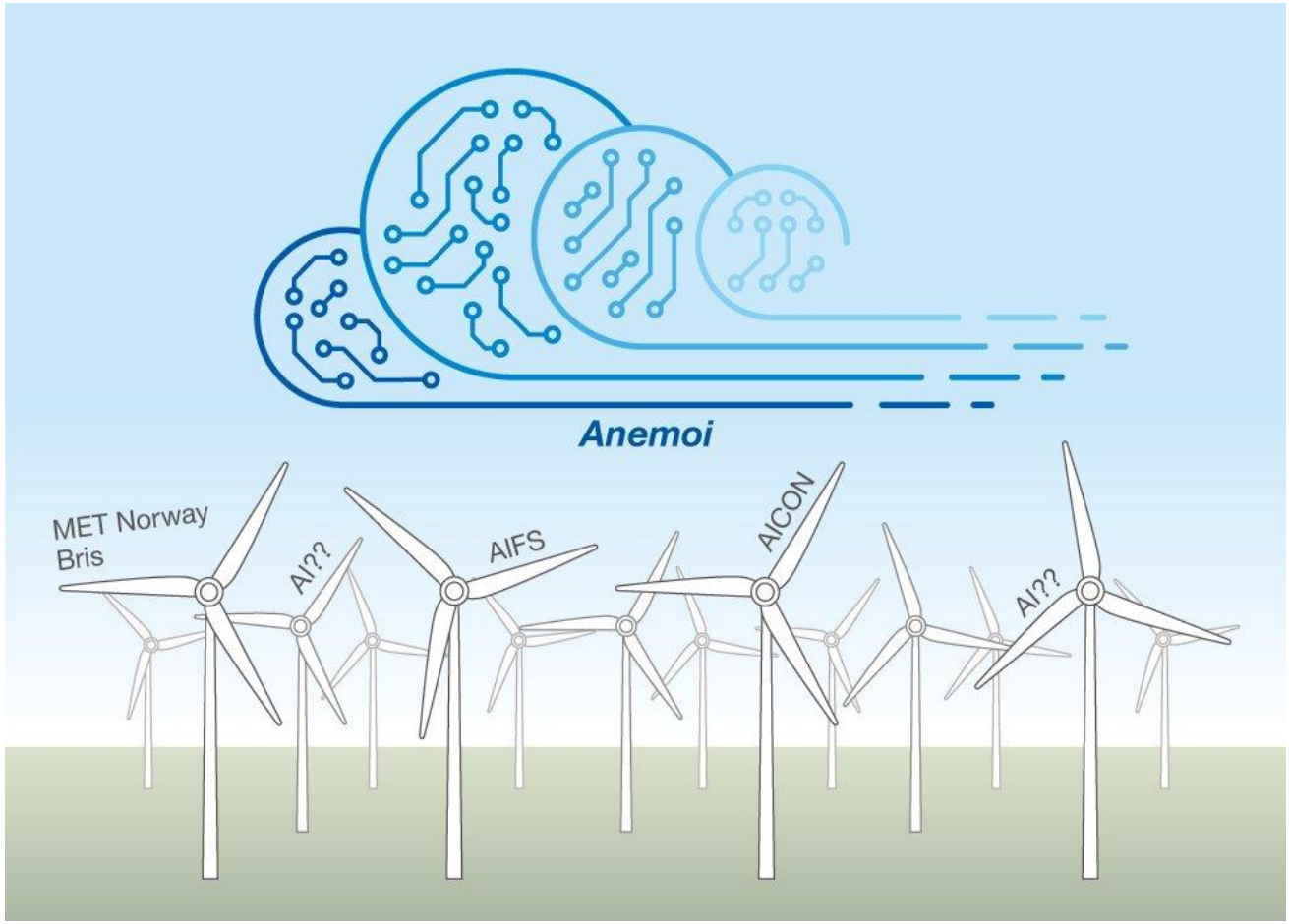
Mario Santa Cruz

Machine Learning Scientist

mario.santacruz@ecmwf.int

Anemoi framework

- Develop machine learning (ML) weather forecasting models.
  - Dataset preparation tools.
  - Graph creation tools.
  - ML model training support.
  - Inference tools integrated with verification software.
  - Registry for datasets and trained models.
- **Benefits:**
  - Simplifies shared meteorological challenges.
  - Enables training of models using existing recipes and custom data.



Anemoi framework

- Develop machine learning (ML) weather forecasting models.
  - Dataset preparation tools.
  - Graph creation tools.
  - ML model training support.
  - Inference tools integrated with verification software.
  - Registry for datasets and trained models.
- **Benefits:**
  - Simplifies shared meteorological challenges.
  - Enables training of models using existing recipes and custom data.

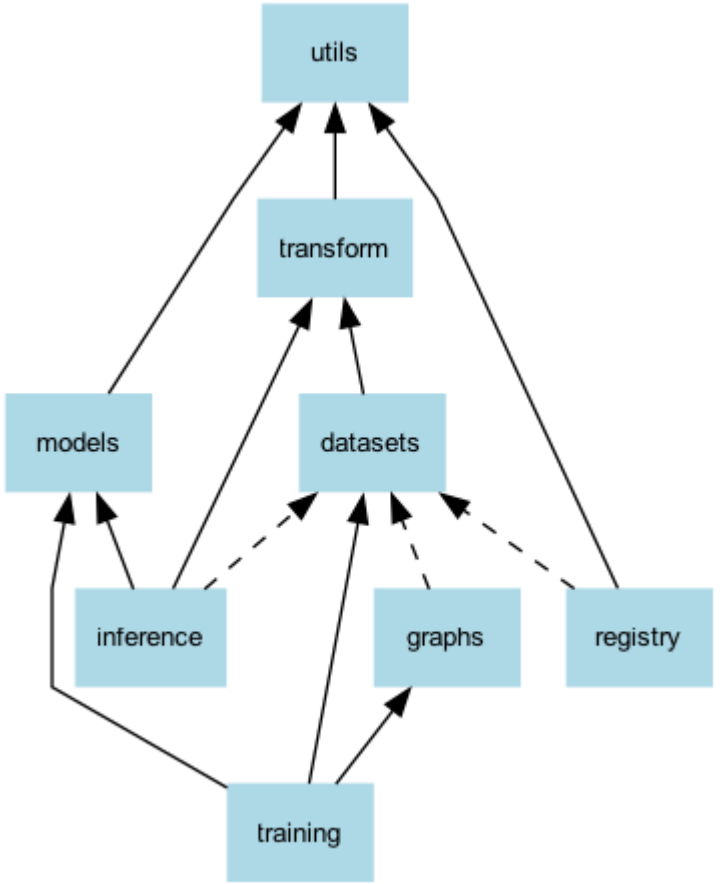
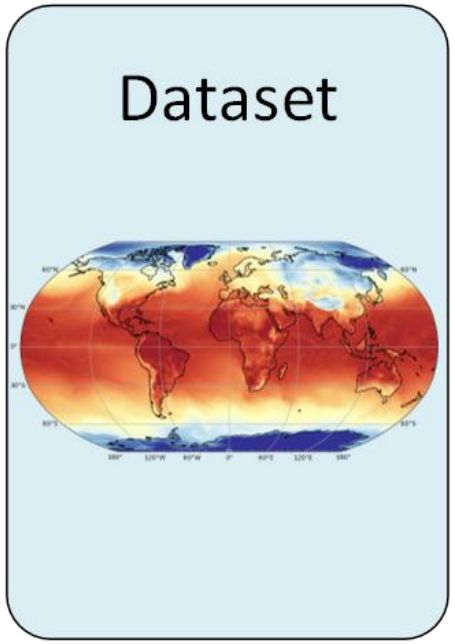
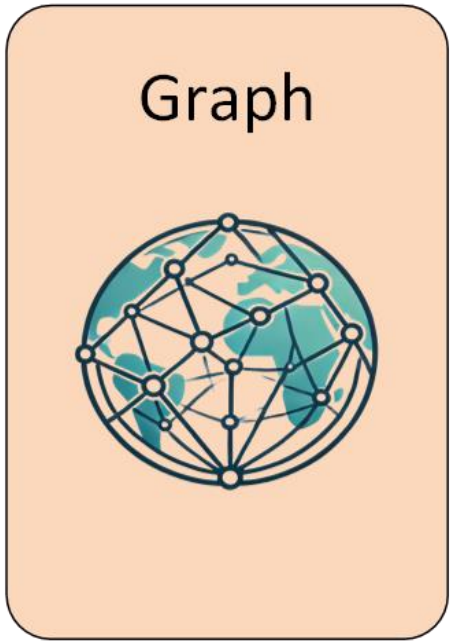


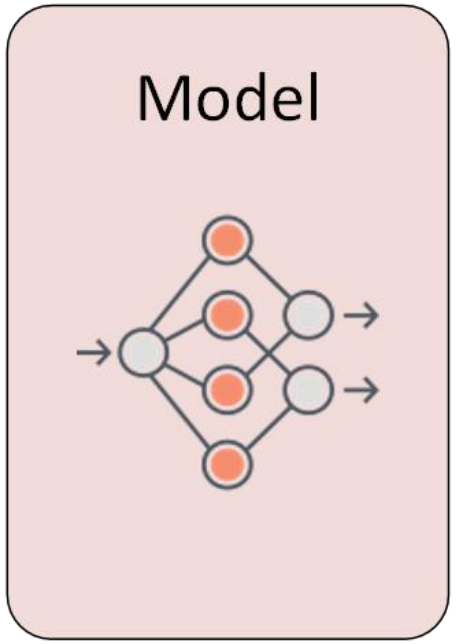
Diagram of dependencies within anemoi packages.

Getting started

To train a data-driven weather forecasting model in **Anemoi**, three main components are needed:

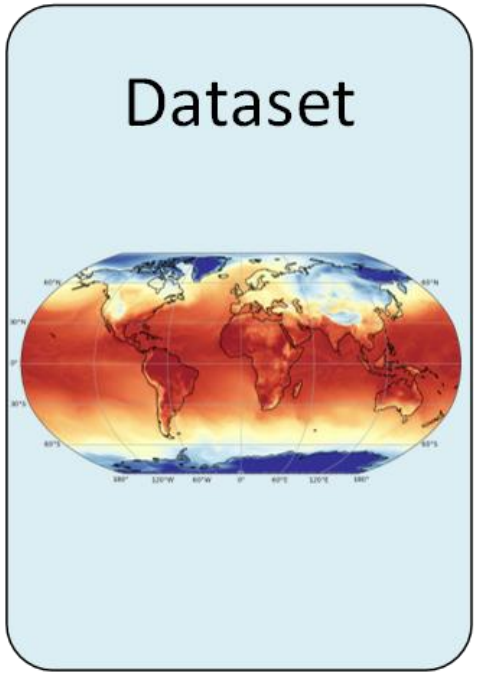




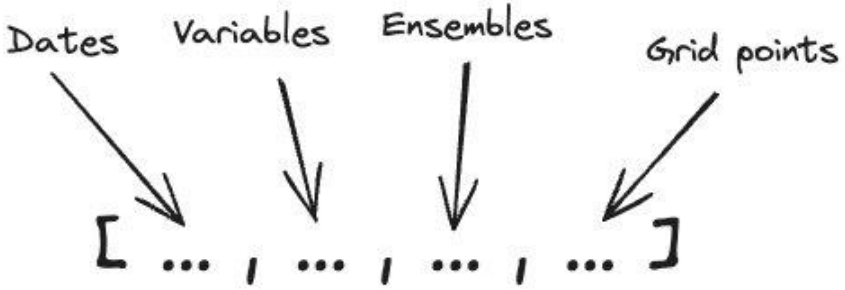


Zarr PyG PyTorch

anemoi-datasets

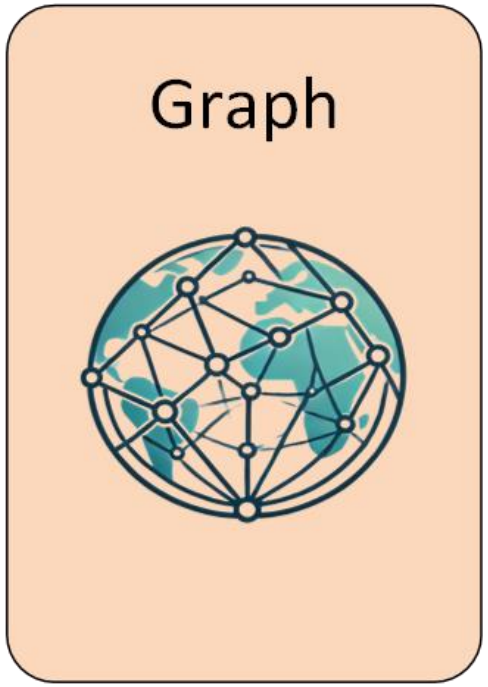


anemoi-datasets.readthedocs.io



- Create "machine-learning ready" datasets for training data-driven weather forecasts.
- Make the loading of data samples as efficient as possible
  - I/O operations are minimised
- Zarr
  - Offers an array-like view on chunks
  - Each file is a single date
- Using datasets
  - Subsetting datasets (time, variables, member, …)
  - Combining datasets (join, concat, cutout, ensemble, ...)

anemoi-graphs



anemoi-graphs.readthedocs.io

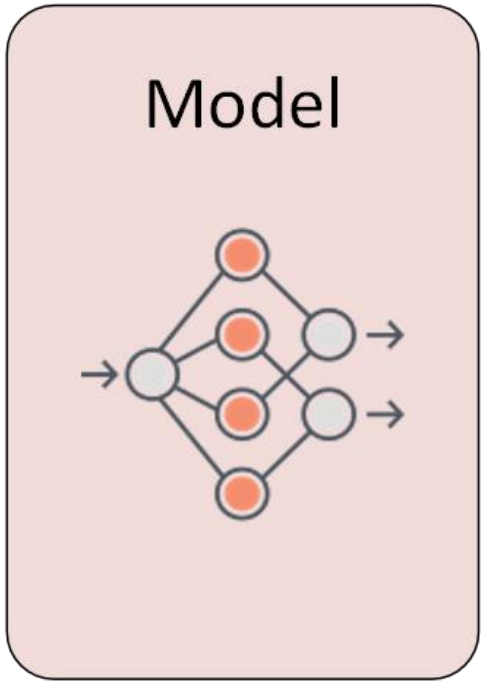
In Anemoi, it is represented by a *torch_geometric.data.**HeteroData*** object, and stored in a .PT file.

- A graph is a structure composed of:
  - **Nodes**: Represent locations in space.
  - **Edges**: Define how information flows between nodes.

```
HeteroData(
  data={
    x=[40320, 2],  # coordinates in radians (lat in [-pi/2, pi/2], lon in [0, 2pi])
    node_type='ZarrDatasetNodes',
    area_weight=[40320, 1],
  },
  hidden={
    x=[10242, 2],  # coordinates in radians (lat in [-pi/2, pi/2], lon in [0, 2pi])
    node_type='TriNodes',
    area_weight=[10242, 1],
  },
  (data, to, hidden)={
    edge_index=[2, 62980],
    edge_type='CutOffEdges',
    edge_length=[62980, 1],
    edge_dirs=[62980, 2],
  },
  (hidden, to, hidden)={
    edge_index=[2, 81900],
    edge_type='MultiScaleEdges',
    edge_length=[81900, 1],
    edge_dirs=[81900, 2],
  },
  (hidden, to, data)={
    edge_index=[2, 120960],
    edge_type='KNNEdges',
    edge_length=[120960, 1],
    edge_dirs=[120960, 2],
  }
)
```

Console log of a graph created with anemoi-graphs.

anemoi-models



anemoi-models.readthedocs.io

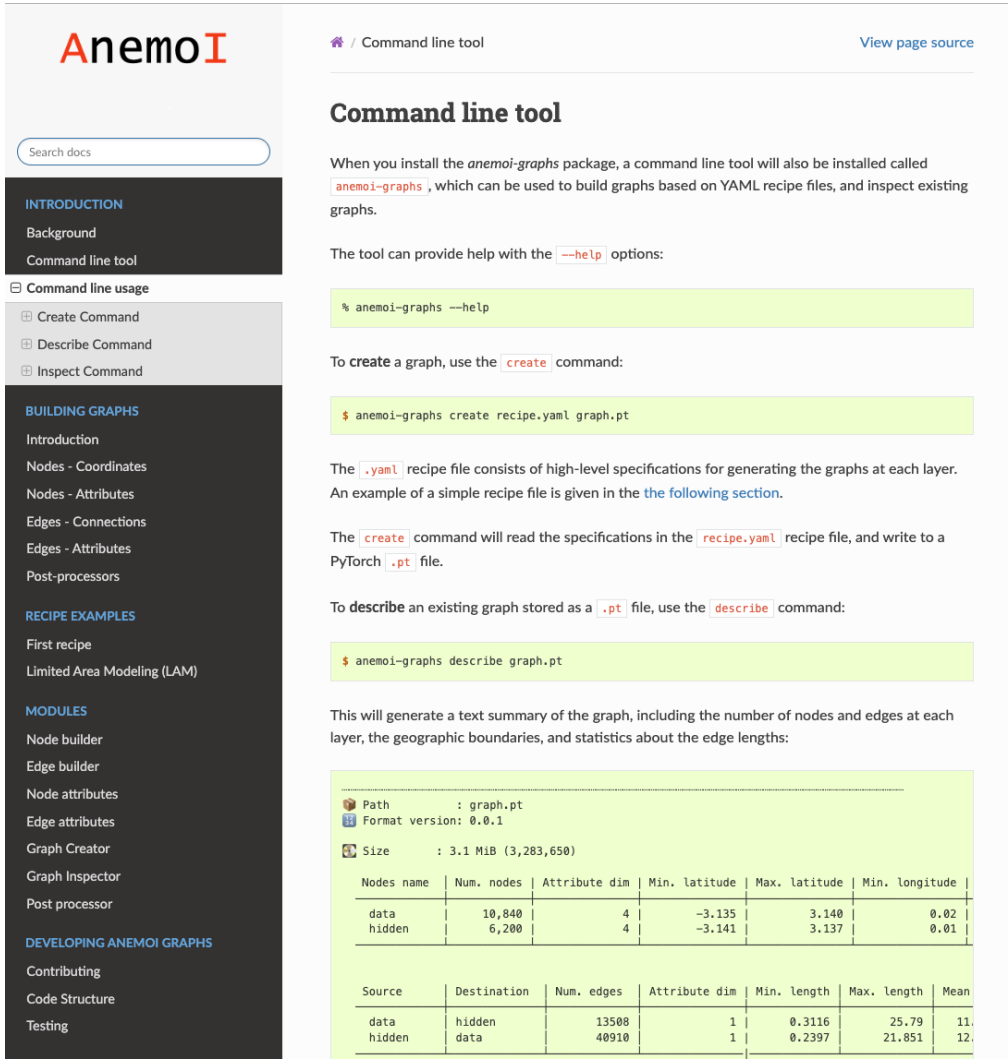
- Graph Neural Networks (GNNs) are a type of neural network designed to operate on **graph-structured** data.
  - GNN (G) = G', where G, G' are graphs
- Node features are update based on *message passing*:
  - Message creation (from node and edge features)
  - Message aggregation
  - Node update
- Permutation-invariance

anemoi-graphs

Tools for creating graphs used in data-driven, deep learning weather forecasting models.

- **High-Level Interface:**
  - **Y**AML recipe file to define graph configuration.
- **Node Definition:**
  - Based on dataset coordinates (e.g., Zarr, NPZ, TXT, ...) or algorithmic methods (e.g., triangular refined icosahedron).
- **Edge Definition:**
  - Methods include cut-off radius or K nearest-neighbors.
- **Attributes:**
  - Supports node and edge attributes (weights, lengths, directions).

anemoi-graphs.readthedocs.io



Documentation of anemoi-graphs package.

Command line tool

- Create a new graph:

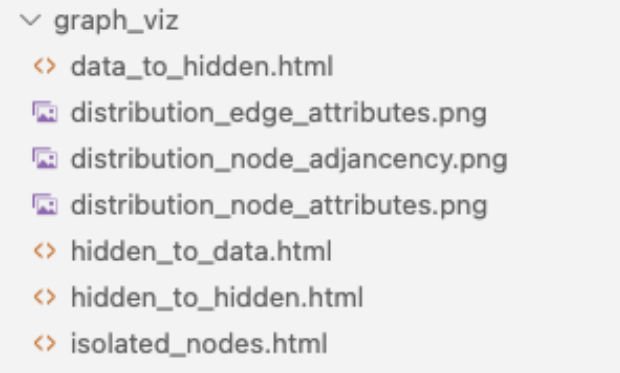
```
>>> anemoi-graphs create recipe.yaml graph.pt
```

- Describe an existing graph:

```
>>> anemoi-graphs describe graph.pt
```

- Inspect visually an existing graph:

```
>>> anemoi-graphs inspect graph.pt graph_viz/
```



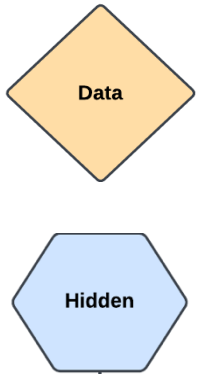
Local files generated to inspect graphs.

```
Path          : graph.pt
Format version: 0.0.1

Size          : 3.1 MiB (3,283,650)

 Nodes name | Num. nodes | Attribute dim | Min. latitude | Max. latitude | Min. longitude |
------------+------------+---------------+---------------+---------------+----------------+
 data       |     10,840 |             4 |        -3.135 |         3.140 |           0.02 |
 hidden     |      6,200 |             4 |        -3.141 |         3.137 |           0.01 |

 Source | Destination | Num. edges | Attribute dim | Min. length | Max. length | Mean
--------+-------------+------------+---------------+-------------+-------------+------
 data   | hidden      |      13508 |             1 |      0.3116 |       25.79 |   11
 hidden | data        |      40910 |             1 |      0.2397 |      21.851 |   12

Graph ready, last update 17 seconds ago.
Statistics ready.
```

Console log when describing/inspecting a graph with anemoi-graphs.

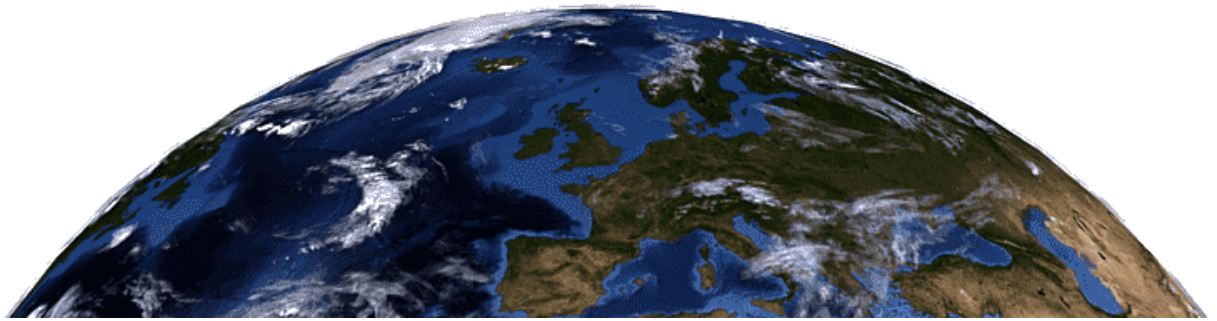
Note: The inspection tools provided are designed for testing different graph configuration but it is not recommendded for high-resolution graphs with a high number of nodes/edges.

Encoder – Processor – Decoder



anemoi-training uses a specific **encoder-processor-decoder** graph structure:

- **Data Nodes**: Represent the locations of input/output data points (e.g., temperature, wind speed).
- **Hidden Nodes**:
  - Operate at a lower spatial resolution than the input/output nodes. (~½)
  - Allow for feature extraction and dimensionality reduction, enabling the model to capture large-scale patterns.

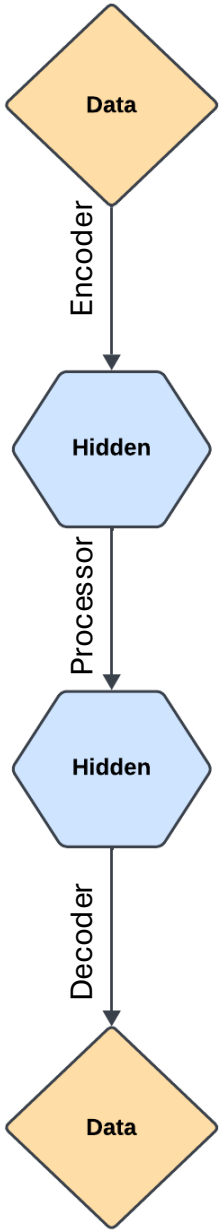


Note: For weather forecasting, the input and output nodes often correspond to the same physical locations. In other use cases (e.g., downscaling), input and output nodes may differ.

Graph recipe

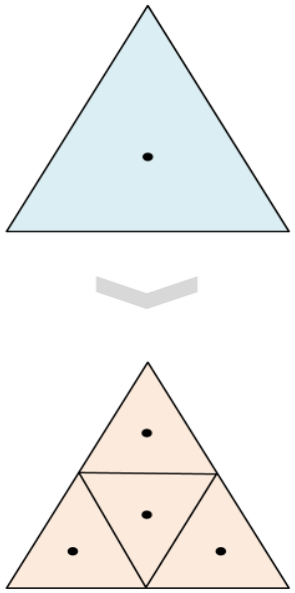
recipe.yaml

```yaml
nodes:
  data:
    node_builder:
      _target_: anemoi.graphs.nodes.AnemoiDatasetNodes
      dataset: my_anemoi_dataset
  hidden:
    node_builder:
      _target_: anemoi.graphs.nodes.TriNodes
      resolution: 5 # num of refinements

edges:
  # Encoder configuration
  - source_name: data
    target_name: hidden
    edge_builders:
      - _target_: anemoi.graphs.edges.CutOffEdges
        cutoff_factor: 0.6
  # Processor configuration
  - source_name: hidden
    target_name: hidden
    edge_builders:
      - _target_: anemoi.graphs.edges.MultiScaleEdges
        x_hops: 1
  # Decoder configuration
  - source_name: hidden
    target_name: data
    edge_builders:
      - _target_: anemoi.graphs.edges.KNNEdges
        num_nearest_neighbours: 3
```

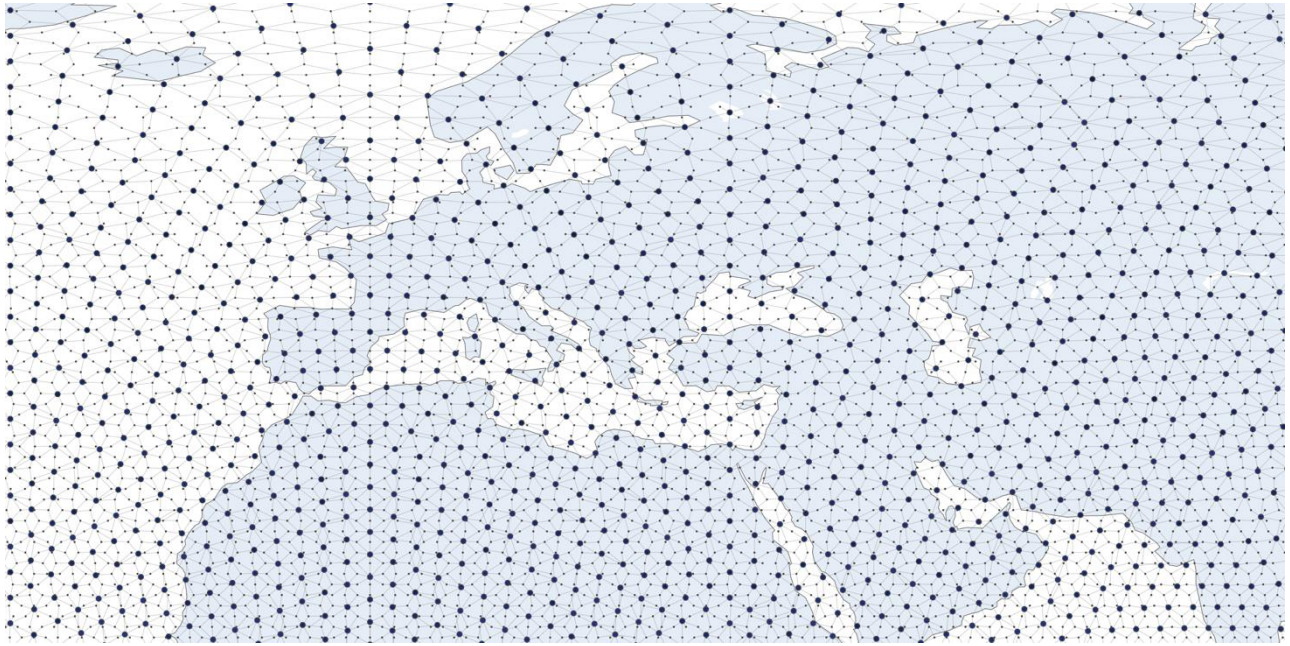


Triangular refined mesh

- It starts with an icosahedron (20 faces) projected to a sphere.
  - Each refinement splits each face into 4 faces.



```
nodes:
  hidden:
    node_builder:
      _target_: anemoi.graphs.nodes.TriNodes
      resolution: 5 # num. of refinements
```



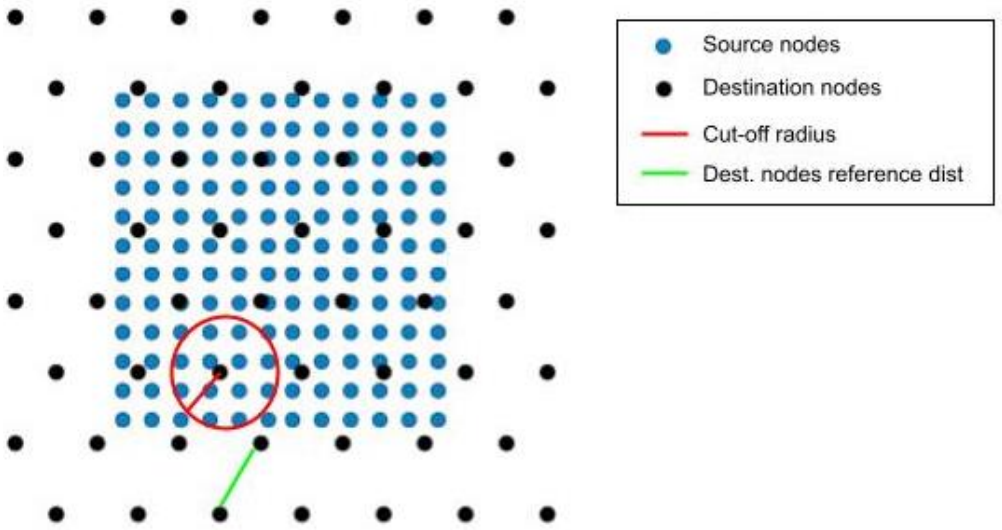
Encoder - CutOffEdges

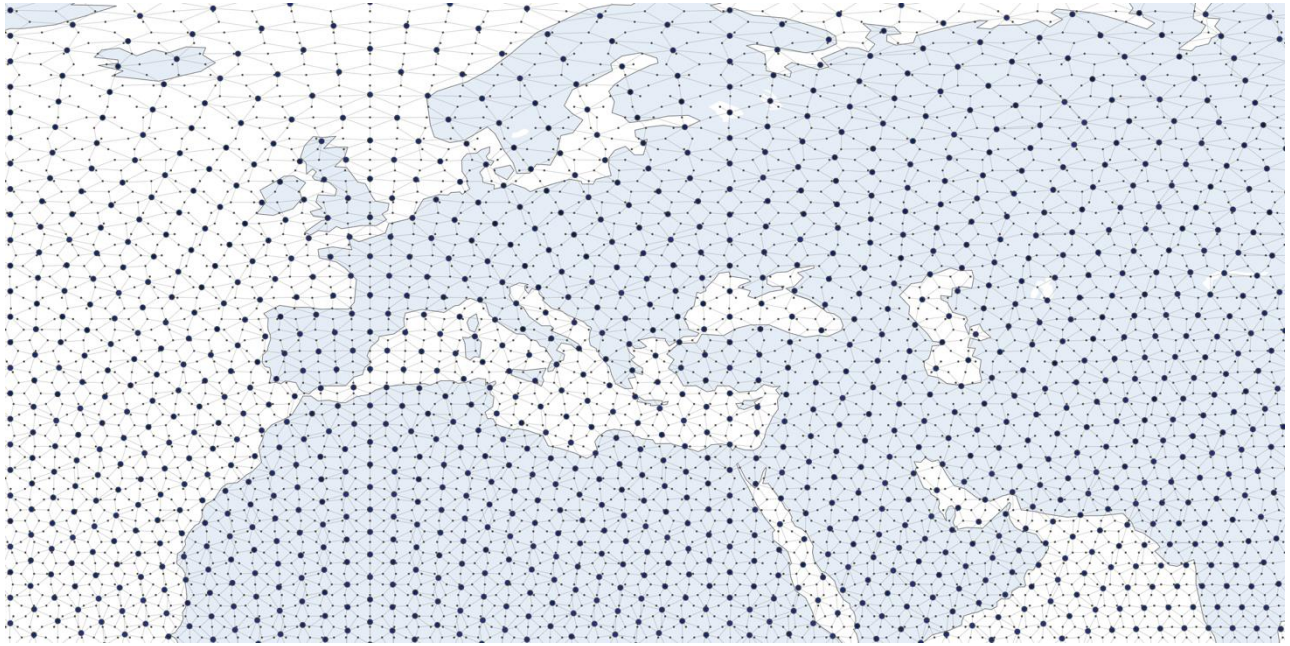
For each target node, it connects all nodes within a given radius.

Why is this important?

- You want to maximise the information flowing to the hidden nodes.
- You want to minimise the number of edges (efficiency)

```
edges:
  - source_name: data
    target_name: hidden
    edge_builders:
      - _target_: anemoi.graphs.edges.CutOffEdges
        x_hops: 1
```

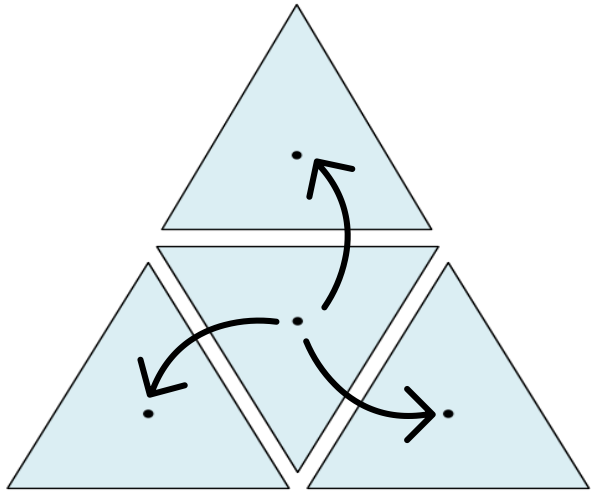


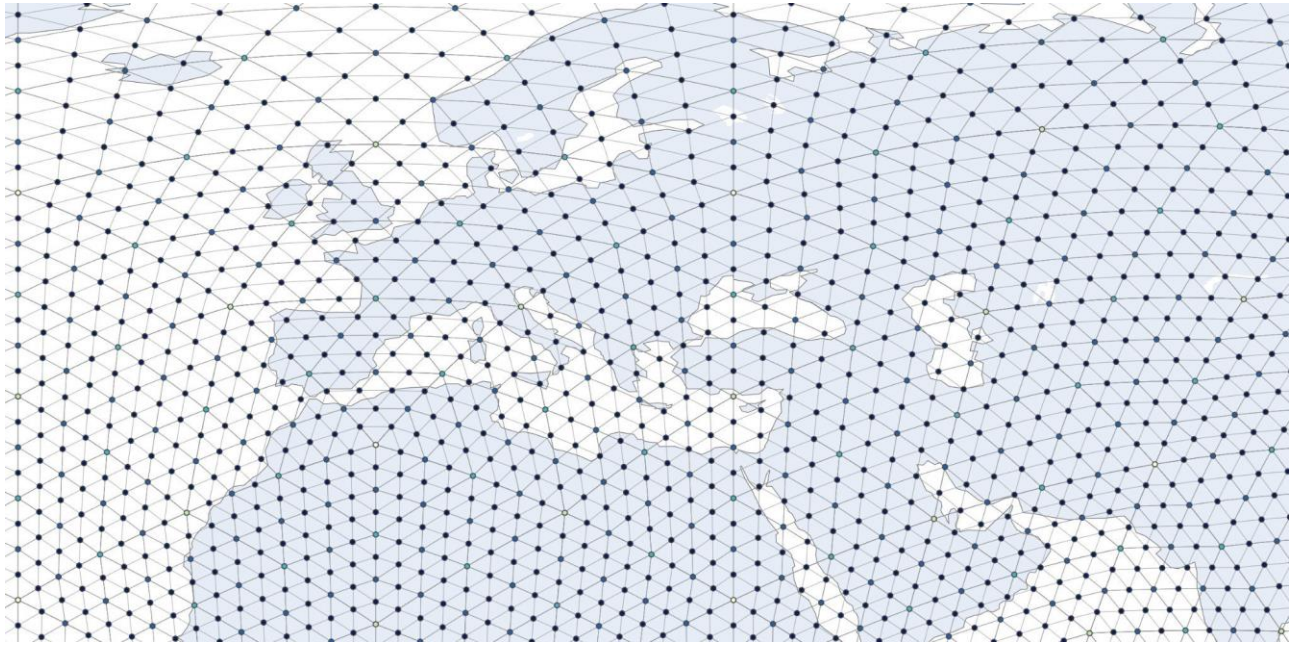


Processor - MultiScaleEdges

- The processor connections define how far the information flows.
  - Connections are created for each level of refinement.

```
edges:
  - source_name: hidden
    target_name: hidden
    edge_builders:
      - _target_: anemoi.graphs.edges.MultiScaleEdges
        x_hops: 1
```

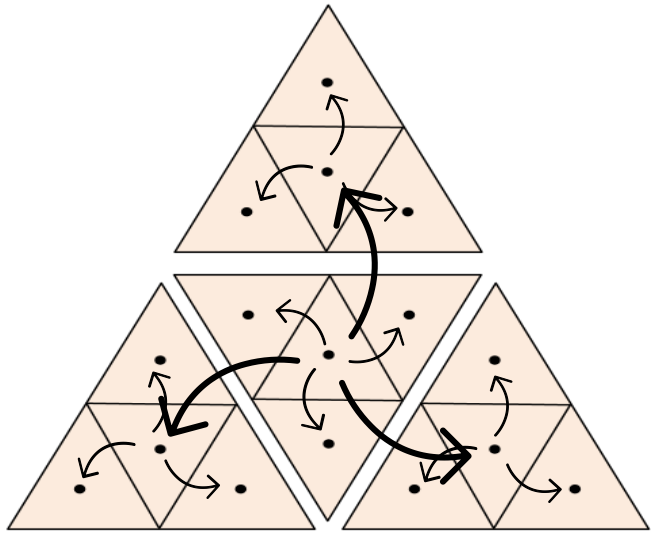


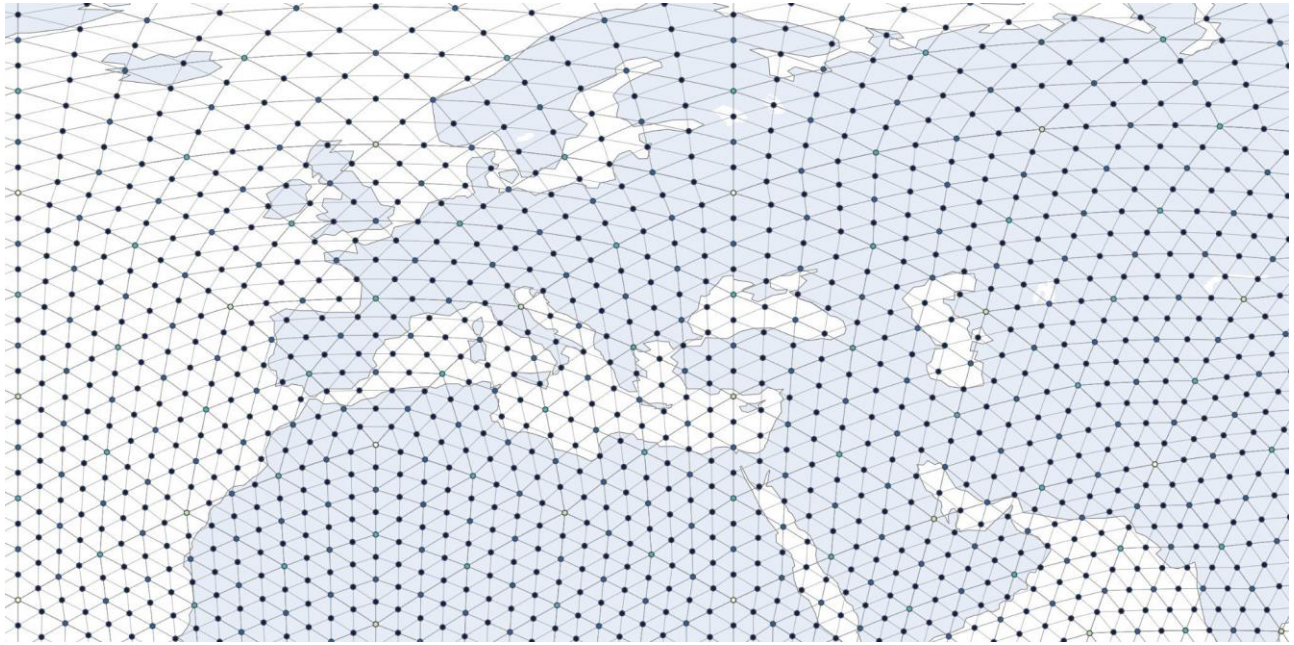


Processor - MultiScaleEdges

- The processor connections define how far the information flows.
  - Connections are created for each level of refinement.

```
edges:
  - source_name: hidden
    target_name: hidden
    edge_builders:
      - _target_: anemoi.graphs.edges.MultiScaleEdges
        x_hops: 1
```

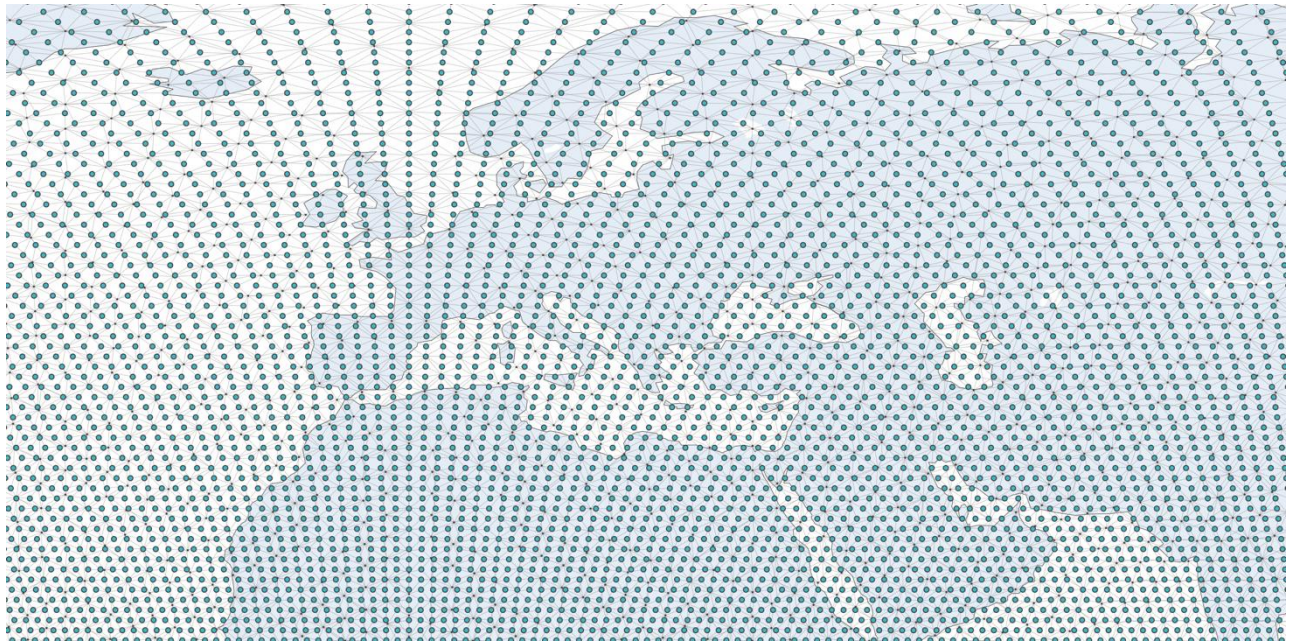




Decoder - KNNEdges

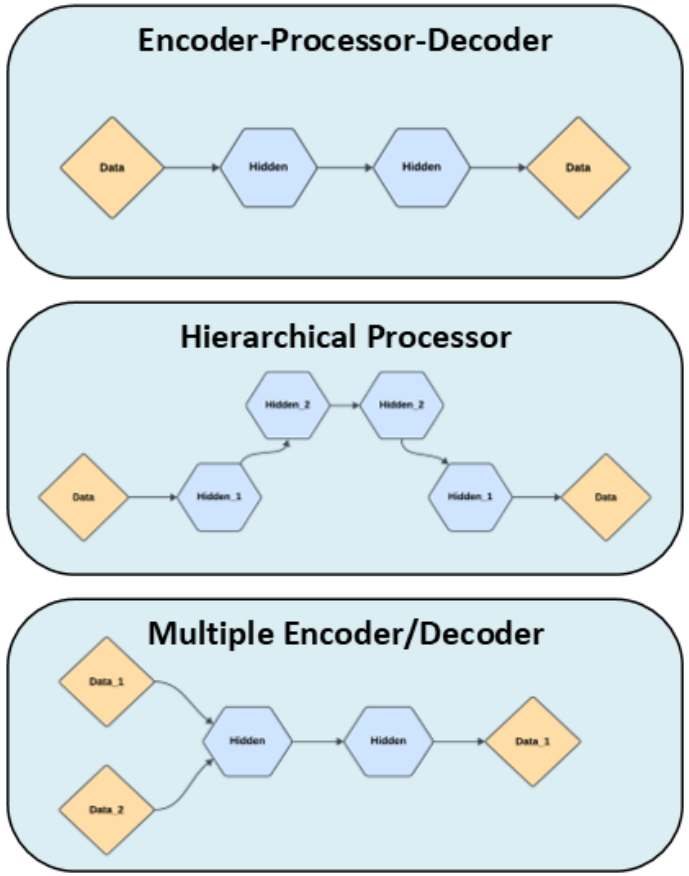
- For each target node, it connects the N nearest neighbours.

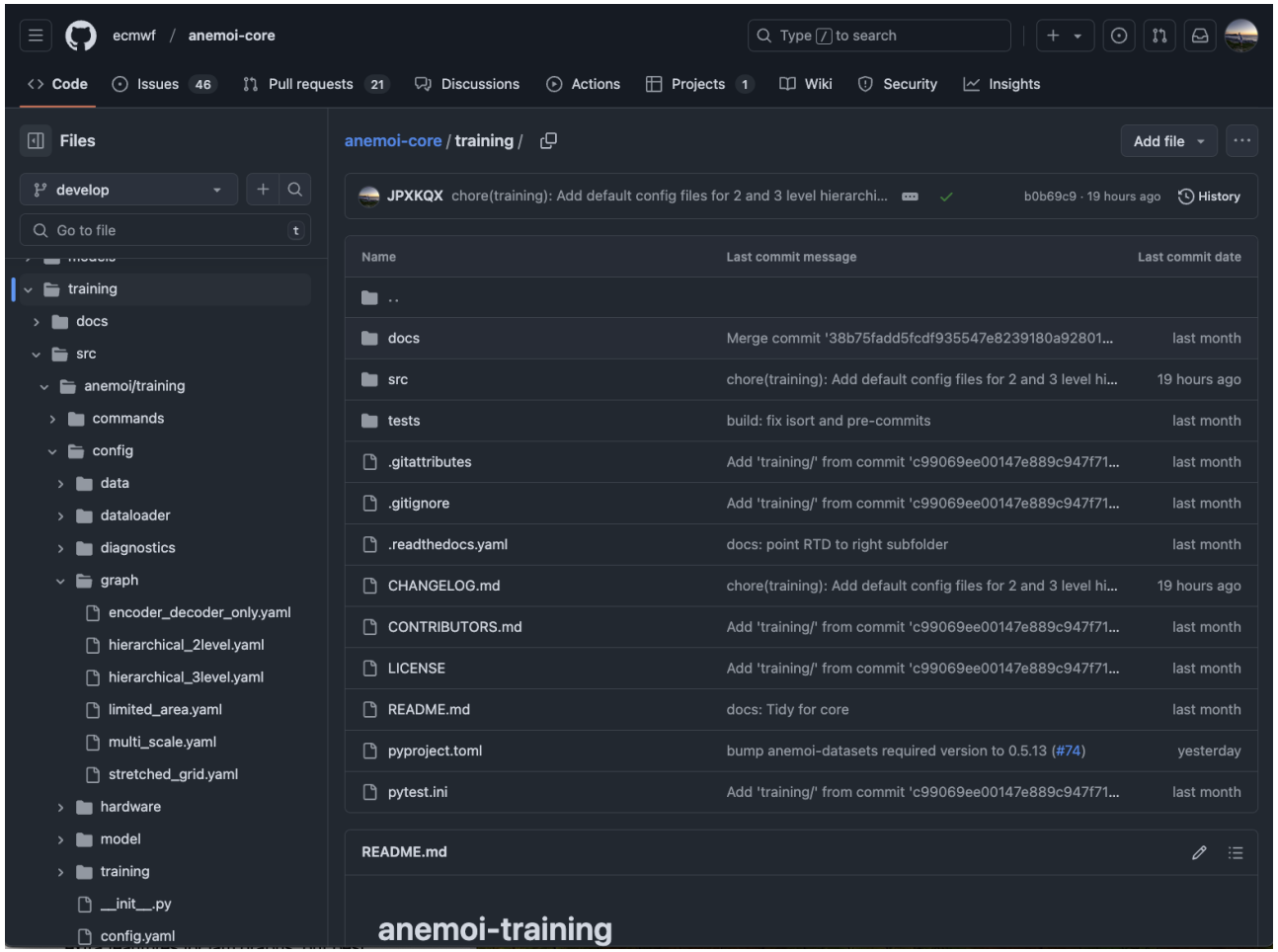
```
edges:
  - source_name: hidden
    target_name: data
    edge_builders:
     - _target_: anemoi.graphs.edges.KNNEdges
       num_nearest_neighbours: 3
```



NOTE: The decoder is the part of the graph with more edges.

Graph configurations



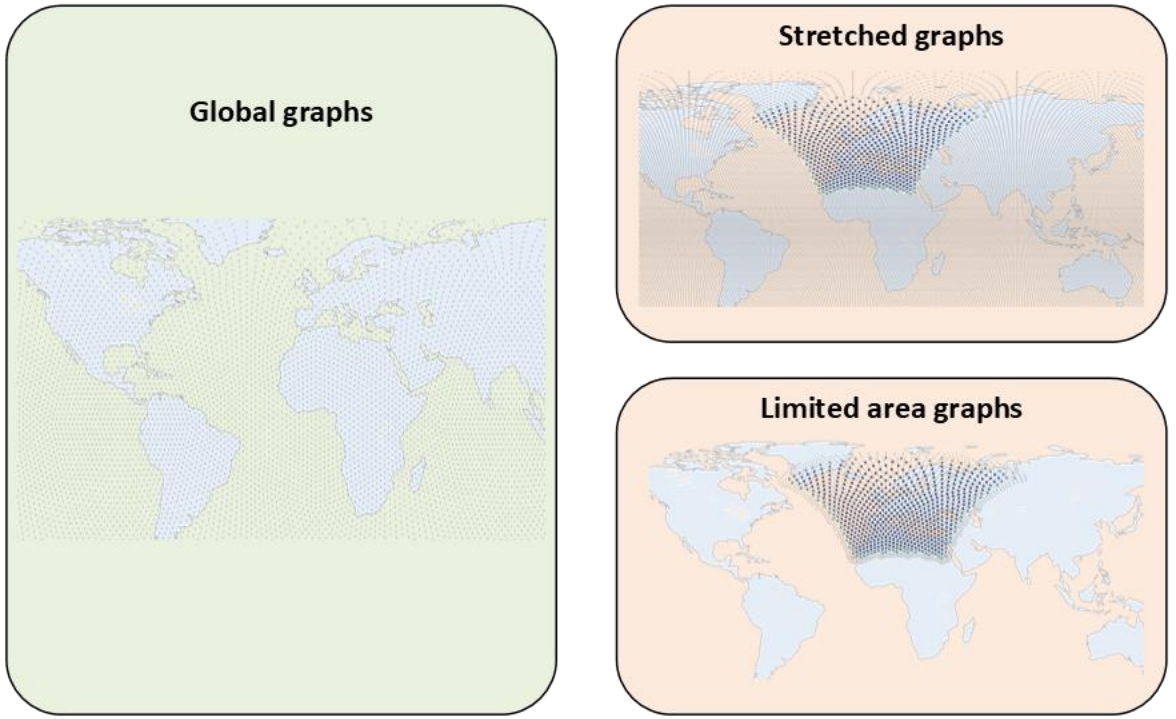


Example configuration files for different use cases.

Use case

Some user may focus in specific regions of interest. In this case, there are several options:

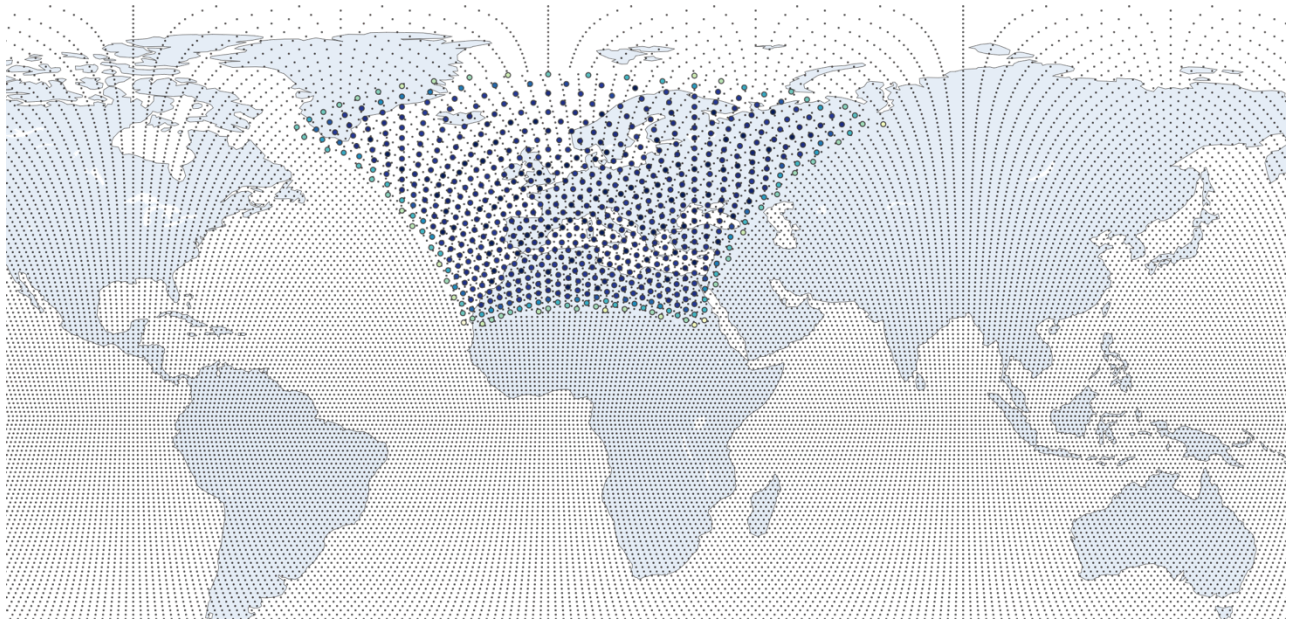
- **Stretched graphs**
  - Increases resolution over a region of interest while maintaining a coarser grid elsewhere.
- **Limited area graphs**
  - Nodes are restricted to a specific region.
  - A coarser dataset can be used as *boundary forcing*.



Regional modelling

```
nodes:
  data:
    node_builder:
      _target_: anemoi.graphs.nodes.ZarrDatasetNodes
      dataset:
        cutout:
        - my_local_zarr_dataset
        - my_global_zarr_dataset
        adjust: all
    attributes:
      area_weight:
        _target_: anemoi.graphs.nodes.attributes.SphericalAreaWeights
        norm: unit-max
      cutout_mask:
        _target_: anemoi.graphs.nodes.attributes.CutOutMask
```

Graph recipe file containing the data nodes configuration.



Data and hidden nodes for a streched graph.

```
hidden:
  node_builder:
    _target_: anemoi.graphs.nodes.LimitedAreaTriNodes
    resolution: 5
    reference_node_name: data
    node_attr_name: cutout_mask
```

Graph recipe file containing the nodes configuration for LAM.

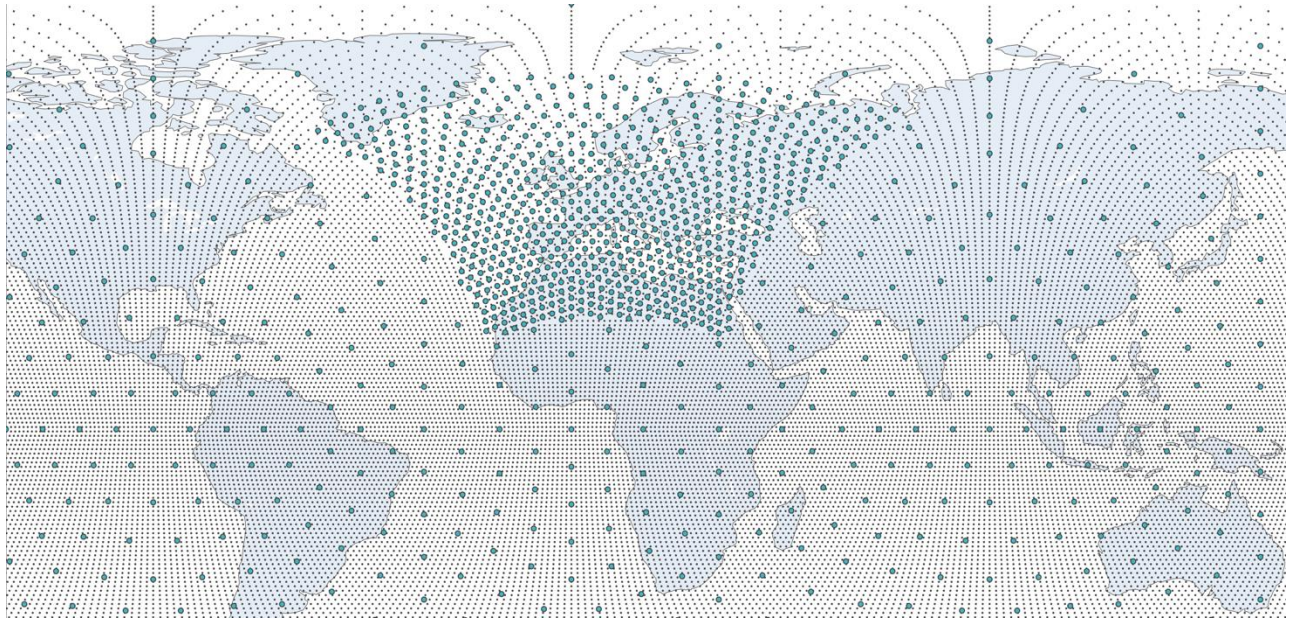
```
hidden:
  node_builder:
    _target_: anemoi.graphs.nodes.StretchedTriNodes
    lam_resolution: 5
    global_resolution: 3
    reference_node_name: data
    node_attr_name: cutout_mask
```

Graph recipe file containing the nodes configuration for stretched graphs.

Regional modelling

```
nodes:
  data:
    node_builder:
      _target_: anemoi.graphs.nodes.ZarrDatasetNodes
      dataset:
        cutout:
        - my_local_zarr_dataset
        - my_global_zarr_dataset
        adjust: all
    attributes:
      area_weight:
        _target_: anemoi.graphs.nodes.attributes.SphericalAreaWeights
        norm: unit-max
      cutout_mask:
        _target_: anemoi.graphs.nodes.attributes.CutOutMask
```

Graph recipe file containing the data nodes configuration.



Data and hidden nodes for a streched graph.

```
hidden:
  node_builder:
    _target_: anemoi.graphs.nodes.LimitedAreaTriNodes
    resolution: 5
    reference_node_name: data
    node_attr_name: cutout_mask
```

Graph recipe file containing the nodes configuration for LAM.

```
hidden:
  node_builder:
    _target_: anemoi.graphs.nodes.StretchedTriNodes
    lam_resolution: 5
    global_resolution: 3
    reference_node_name: data
    node_attr_name: cutout_mask
```

Graph recipe file containing the nodes configuration for stretched graphs.

Limited Area Modelling

```
edges:
  # Encoder configuration
  - source_name: data
    target_name: hidden
    edge_builders:
      - _target_: anemoi.graphs.edges.CutOffEdges
        cutoff_factor: 0.6
    attributes:
      edge_length:
        _target_: anemoi.graphs.edges.attributes.EdgeLength
  # Processor configuration
  - source_name: hidden
    target_name: hidden
    edge_builders:
      - _target_: anemoi.graphs.edges.MultiScaleEdges
        x_hops: 1
    attributes:
      edge_length:
        _target_: anemoi.graphs.edges.attributes.EdgeLength
  # Decoder configuration
  - source_name: hidden
    target_name: data
    edge_builders:
      - _target_: anemoi.graphs.edges.KNNEdges
        target_mask_attr_name: cutout_mask
        num_nearest_neighbours: 3
    attributes:
      edge_length:
        _target_: anemoi.graphs.edges.attributes.EdgeLength
```

Cont.: Graph recipe file containing the edge configuration for LAM.

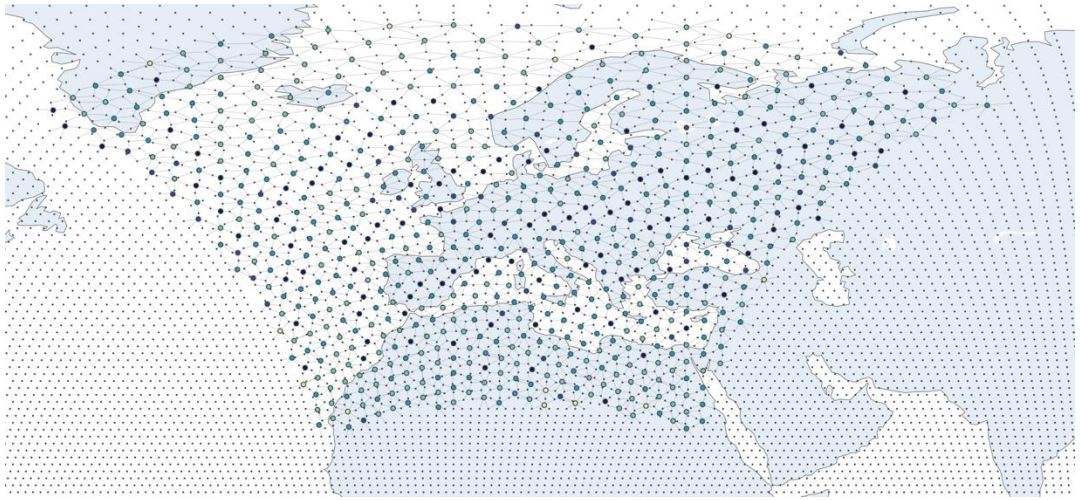


Diagram of encoder connections.

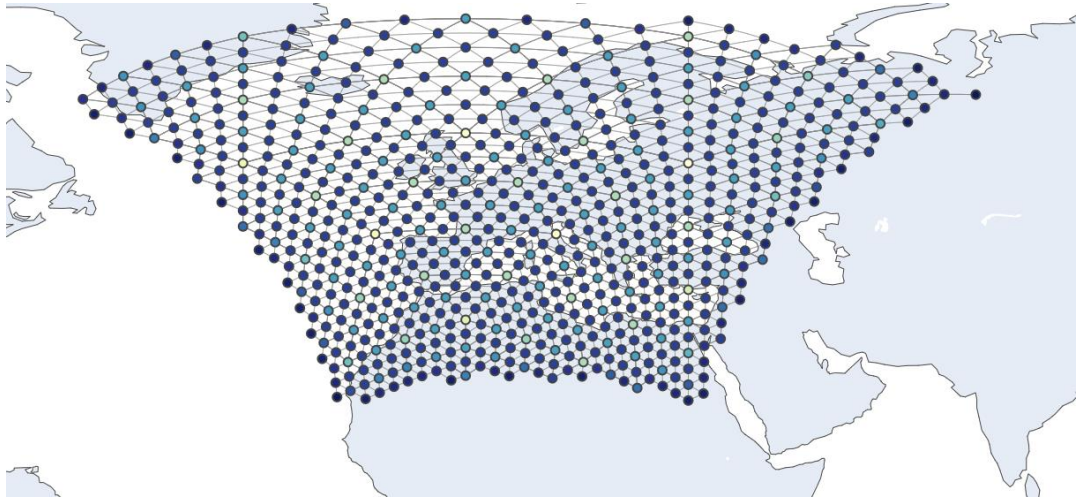


Diagram of processor connections.

Wrap up

- *anemoi-graphs* is used by *anemoi-training* to create the graphs on the fly.
- *anemoi-training* contains recipes for all use cases
- *anemoi-graphs* has a command line tool to create and inspect graphs
- *anemoi-graphs* is designed to be extended with new node, edge and attribute builders.
- *anemoi-graphs* supports different use cases:
  - Global graphs
  - Limited area graphs
  - Stretched graphs
- More complex setups (multi-encoder/decoder, hierarchi l graphs, …) are supported in *anemoi-graphs*.
- Graph configuration plays a key role in the flow of information and the efficiency of the model.

Questions?

anemoi-graphs.readthedocs.io

mario.santacruz@ecmwf.int

www.ecmwf.int

DISCOVER ANEMOI · WEBINAR SERIES · PART 2

Anemoi models

PRESENTER

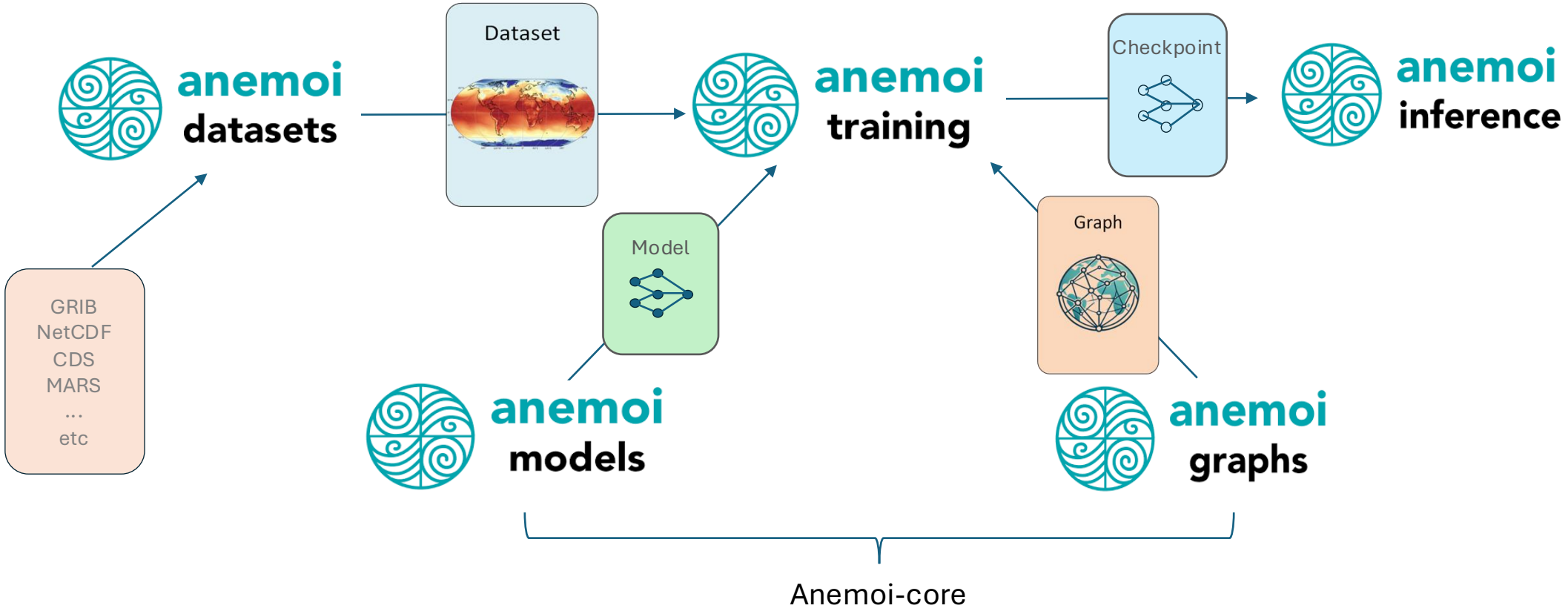
Mario Santa Cruz

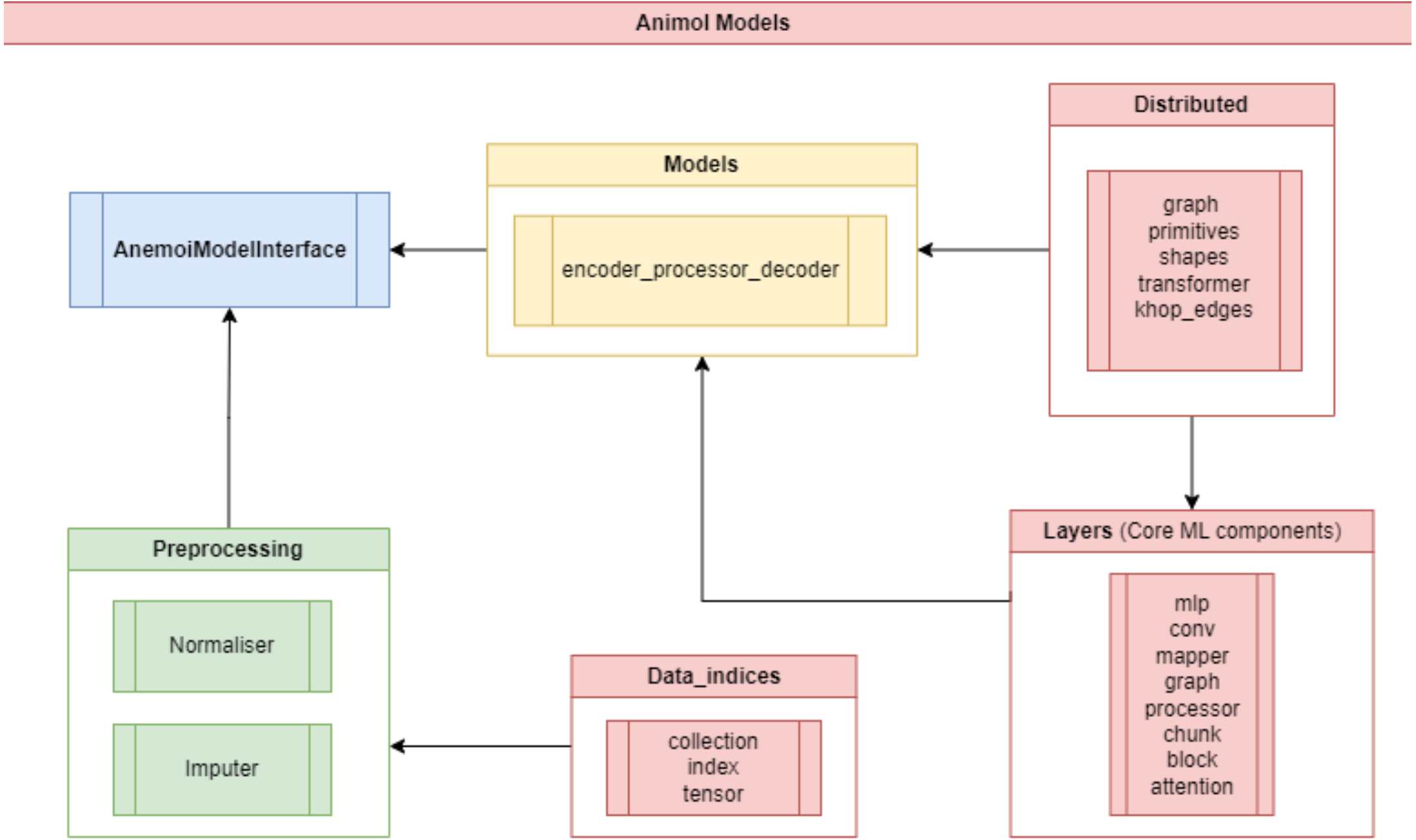
Machine Learning Scientist

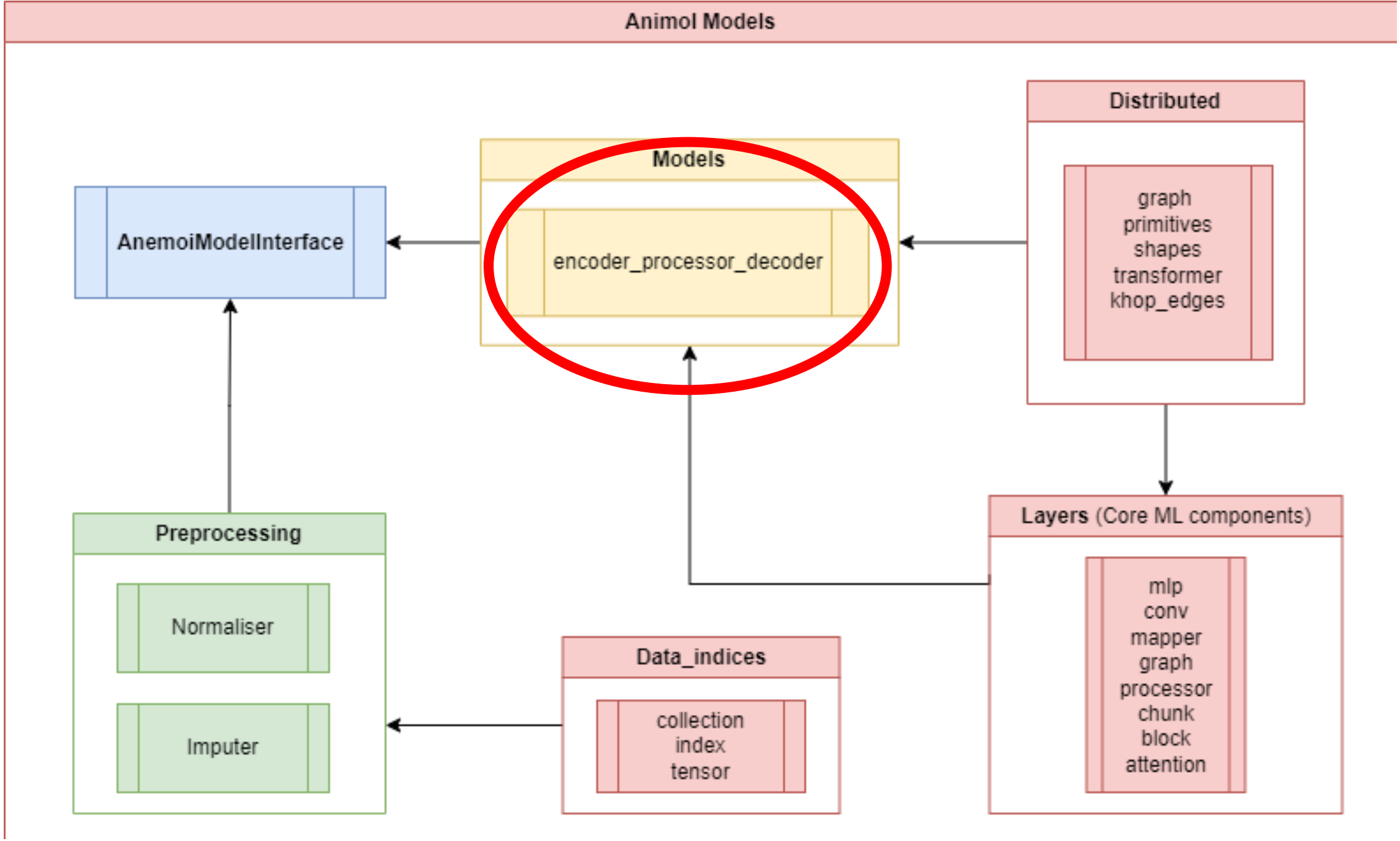
mario.santacruz@ecmwf.int

Anemoi in a Nutshell

- Anemoi is an open-source framework to develop **data-driven meteorological forecasting**.
- Anemoi is developed through a **collaborative European initiative**

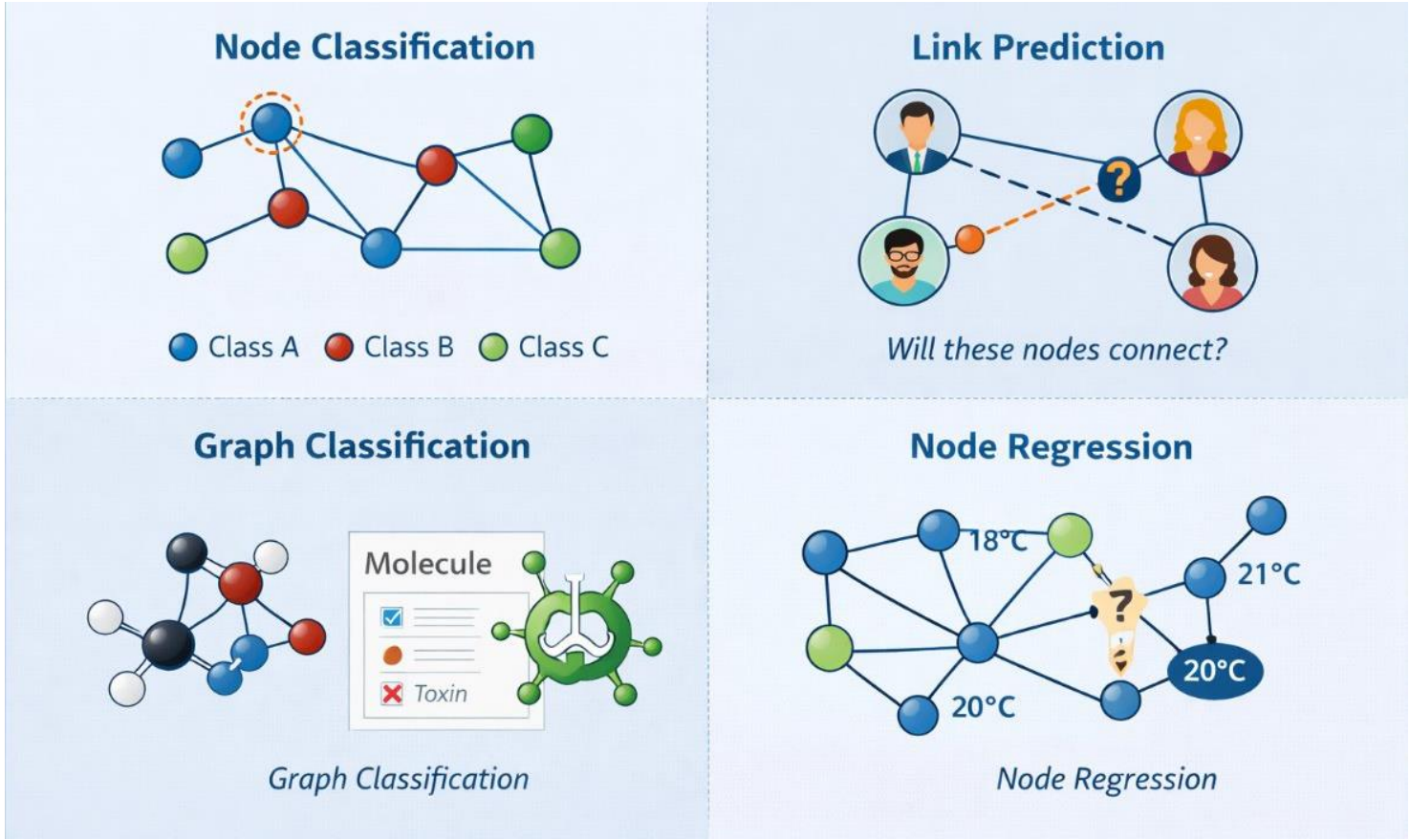



Animol Models
Distributed
graph
primitives
shapes
transformer
khop_edges
Models
encoder_processor_decoder
AnemoiModelInterface
Preprocessing
Normaliser
Imputer
Data_indices
collection
index
tensor
Layers (Core ML components)
mlp
conv
mapper
graph
processor
chunk
block
attention


Animol Models
Distributed
graph
primitives
shapes
transformer
khop_edges
Models
encoder_processor_decoder
AnemoiModelInterface
Preprocessing
Normaliser
Imputer
Data_indices
collection
index
tensor
Layers (Core ML components)
mlp
conv
mapper
graph
processor
chunk
block
attention

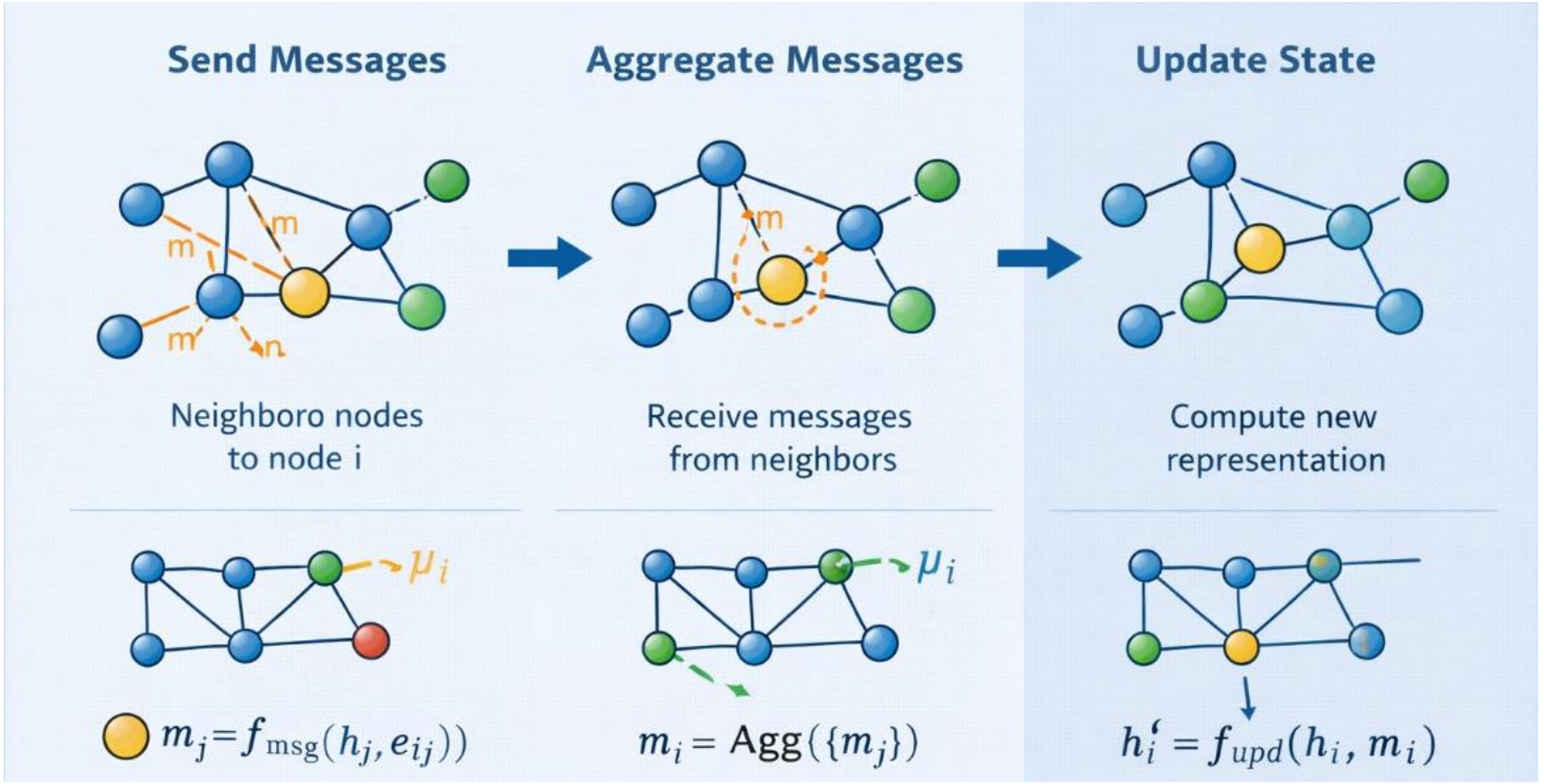
Graph Neural Networks

- Graph Neural Networks (GNNs) are a type of neural network designed to operate on **graph-structured** data.
    - GNN (G) = G', where G, G' are graphs

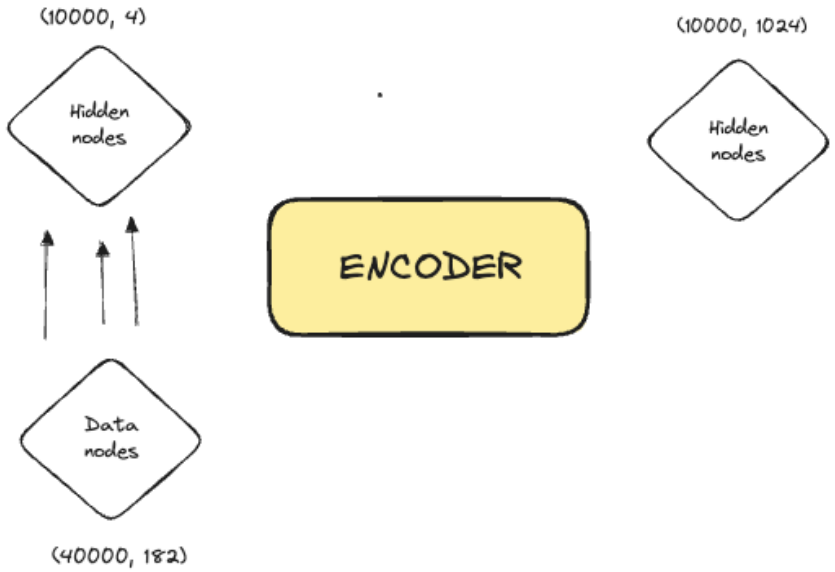


Graph Neural Networks

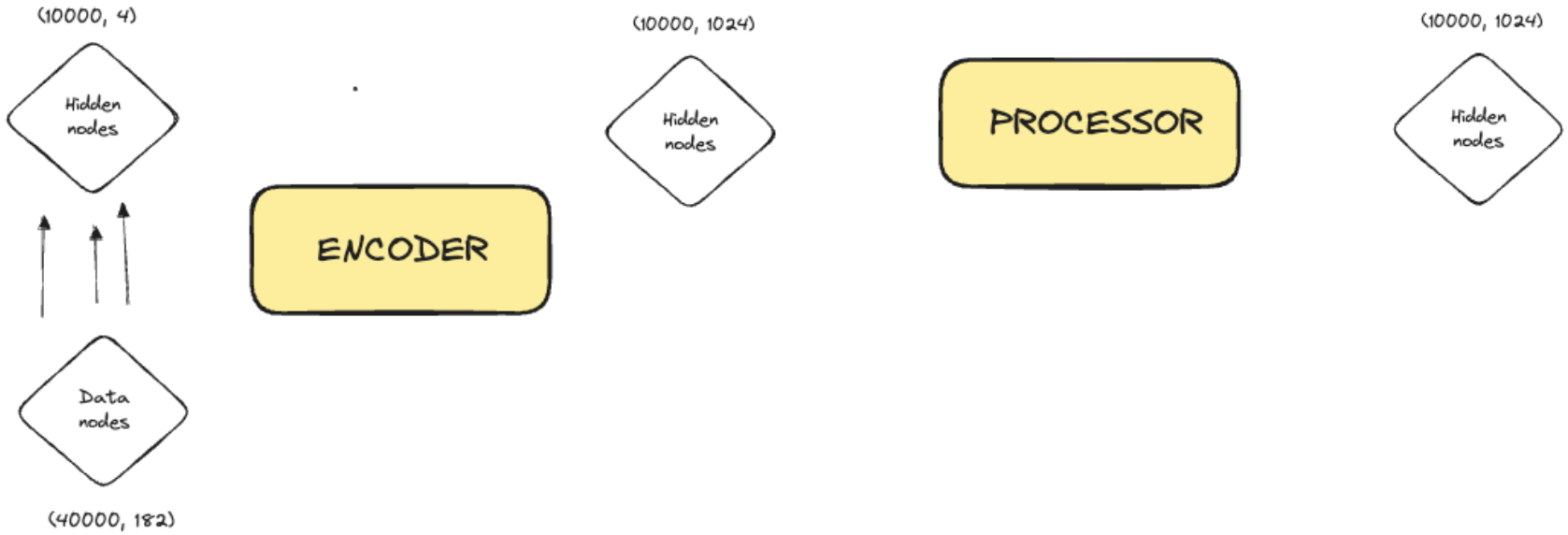
- Node features are update based on *message passing*:
  - Message creation (from node and edge features)
  - Message aggregation
  - Node update



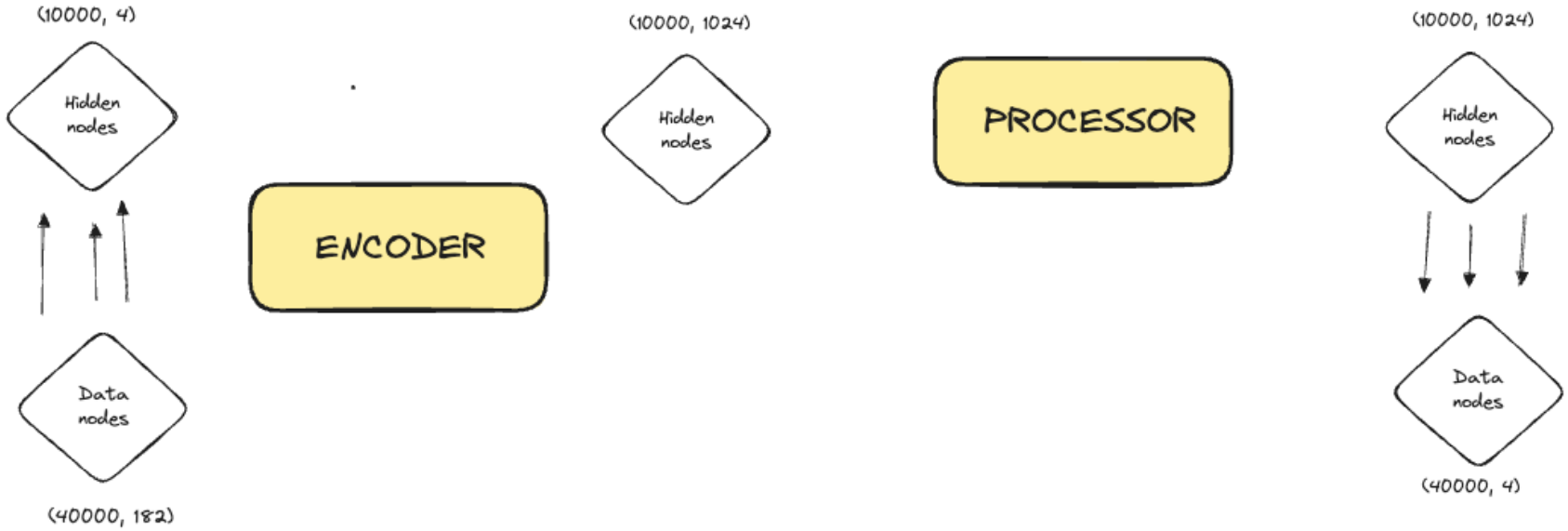
Encoder-Processor-Decoder



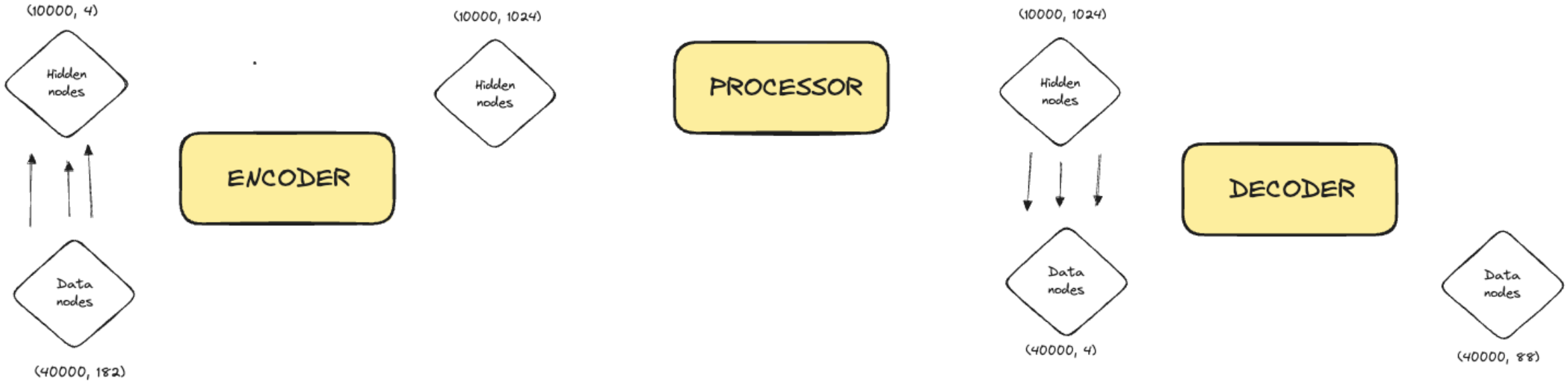
Encoder-Processor-Decoder

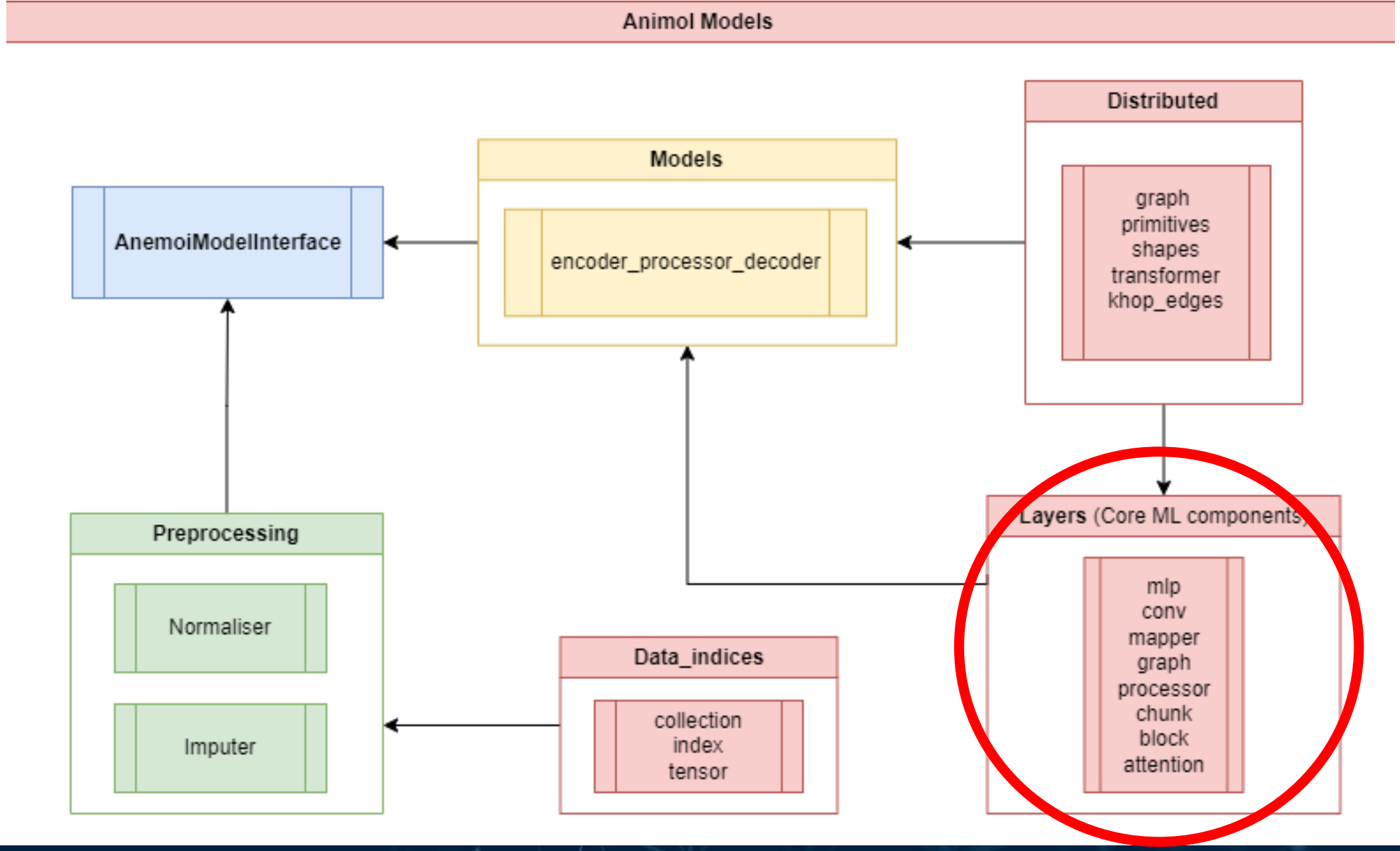


Encoder-Processor-Decoder



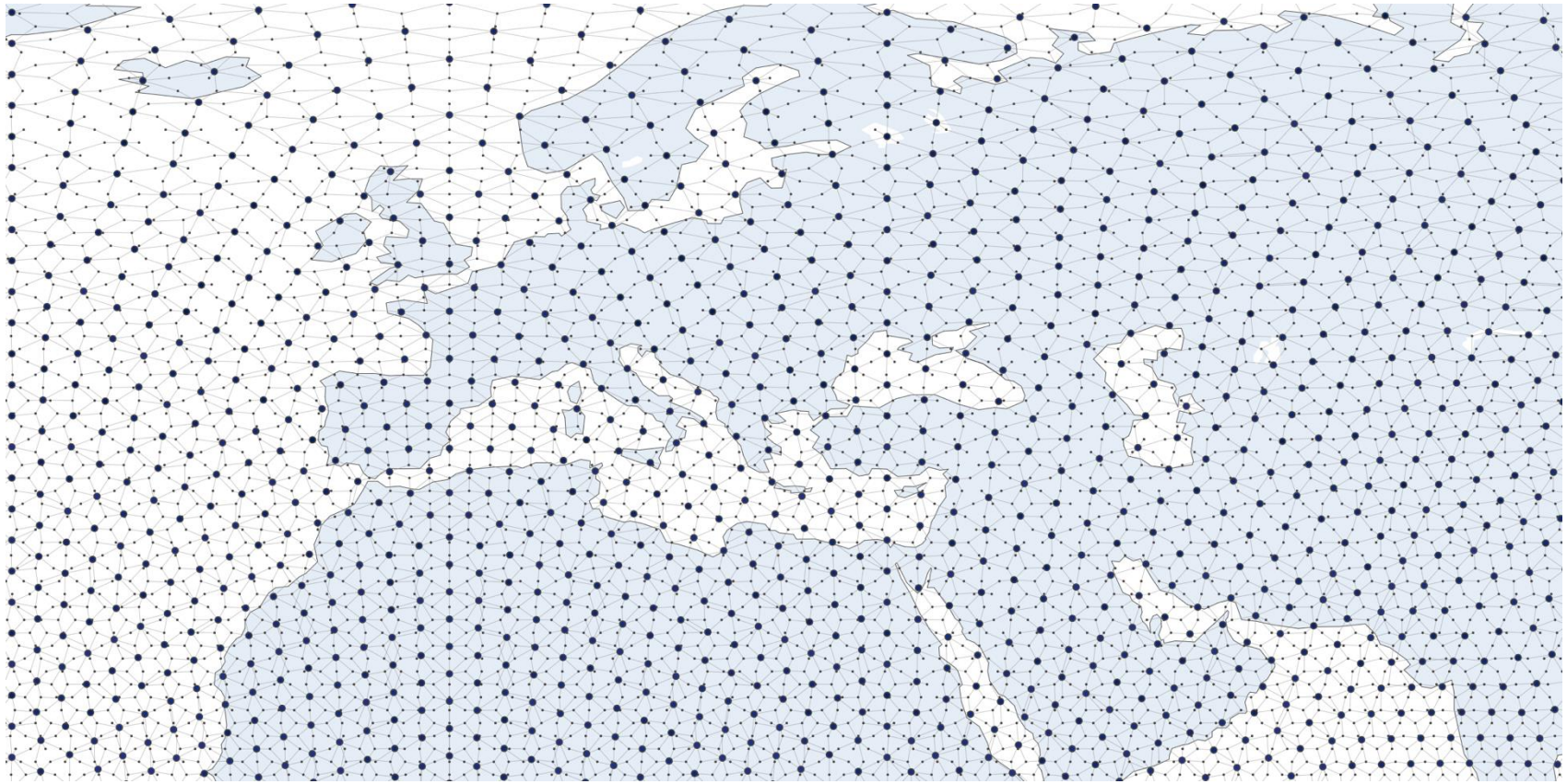
Encoder-Processor-Decoder




Animol Models
Distributed
graph
primitives
shapes
transformer
khop_edges
Models
encoder_processor_decoder
AnemoiModelInterface
Layers (Core ML components)
mlp
conv
mapper
graph
processor
chunk
block
attention
Preprocessing
Normaliser
Imputer
Data_indices
collection
index
tensor

How to build an encoder processor decoder?

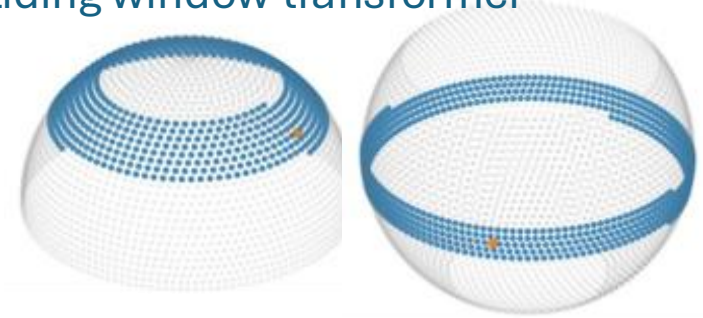
- Encoders/decoders use Mappers.
  - Map from one grid to another.
  - 4 to choose from:
    - GNN (message passing)
    - **GraphTransformer**
    - Transformer
    - PointWiseMLP



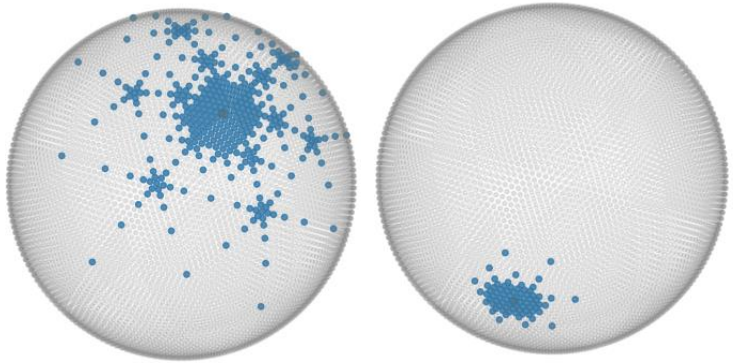
How to build an encoder processor decoder?

- Processors layers to build the processor.
  - Operate on fixed grid.
  - 4 to choose from:
    - GNN (message passing)
    - **GraphTransformer**
    - **Transformer**
    - PointWiseMLP

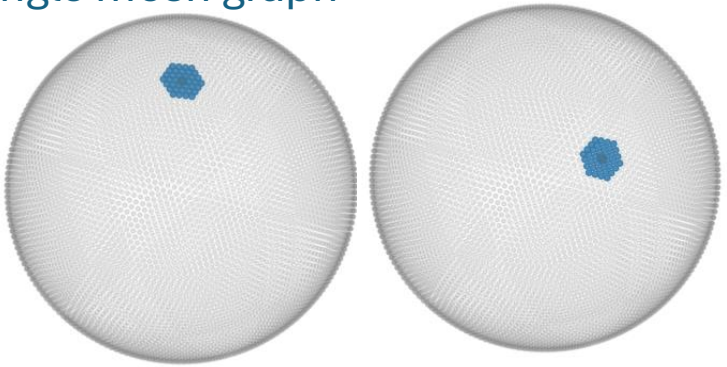
Sliding window transformer



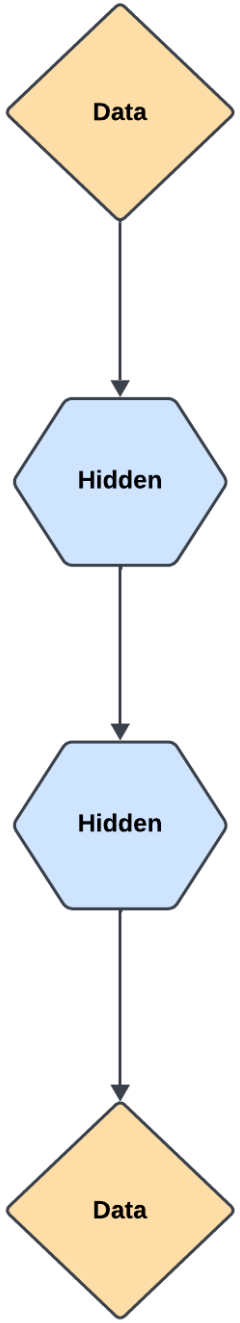
Multi mesh graph

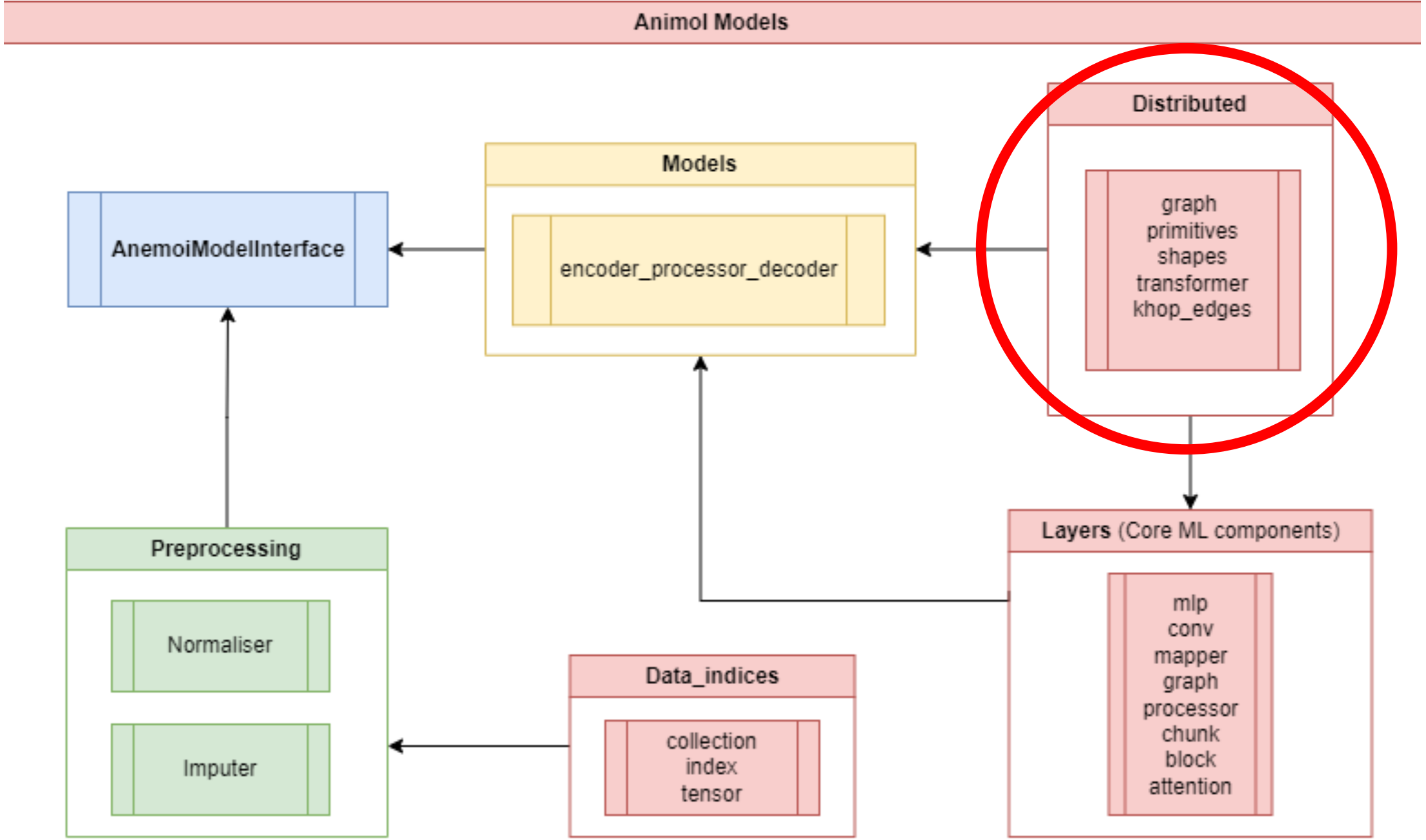


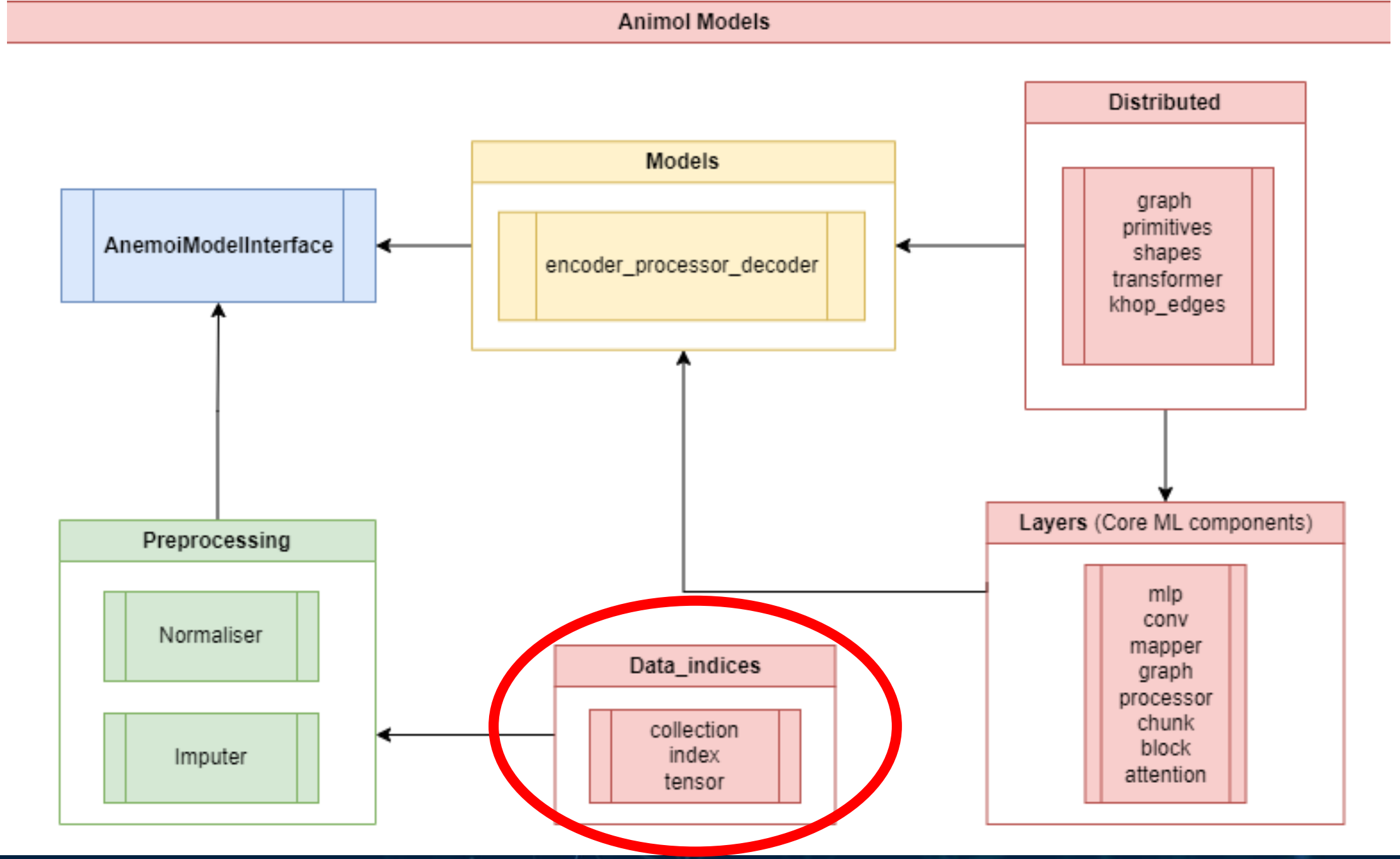
Single mesh graph



```yaml
encoder:
  _target_: anemoi.models.layers.mapper.GraphTransformerForwardMapper
  trainable_size: ${model.trainable_parameters.data2hidden}
  sub_graph_edge_attributes: ${model.attributes.edges}
  num_chunks: 4
  mlp_hidden_ratio: 4 # GraphTransformer or Transformer only
  num_heads: 16 # GraphTransformer or Transformer only
  qk_norm: False
  cpu_offload: ${model.cpu_offload}
  layer_kernels: ${model.layer_kernels}
  shard_strategy: "edges"
processor:
  _target_: anemoi.models.layers.processor.GraphTransformerProcessor
  trainable_size: ${model.trainable_parameters.hidden2hidden}
  sub_graph_edge_attributes: ${model.attributes.edges}
  num_layers: 16
  num_chunks: 4
  mlp_hidden_ratio: 4 # GraphTransformer or Transformer only
  num_heads: 16 # GraphTransformer or Transformer only
  qk_norm: False
  cpu_offload: ${model.cpu_offload}
  layer_kernels: ${model.layer_kernels}
decoder:
  _target_: anemoi.models.layers.mapper.GraphTransformerBackwardMapper
  trainable_size: ${model.trainable_parameters.hidden2data}
  sub_graph_edge_attributes: ${model.attributes.edges}
  num_chunks: 4
  mlp_hidden_ratio: 4 # GraphTransformer or Transformer only
  num_heads: 16 # GraphTransformer or Transformer only
  initialise_data_extractor_zero: False
  qk_norm: False
  cpu_offload: ${model.cpu_offload}
  layer_kernels: ${model.layer_kernels}
  shard_strategy: "edges"
```

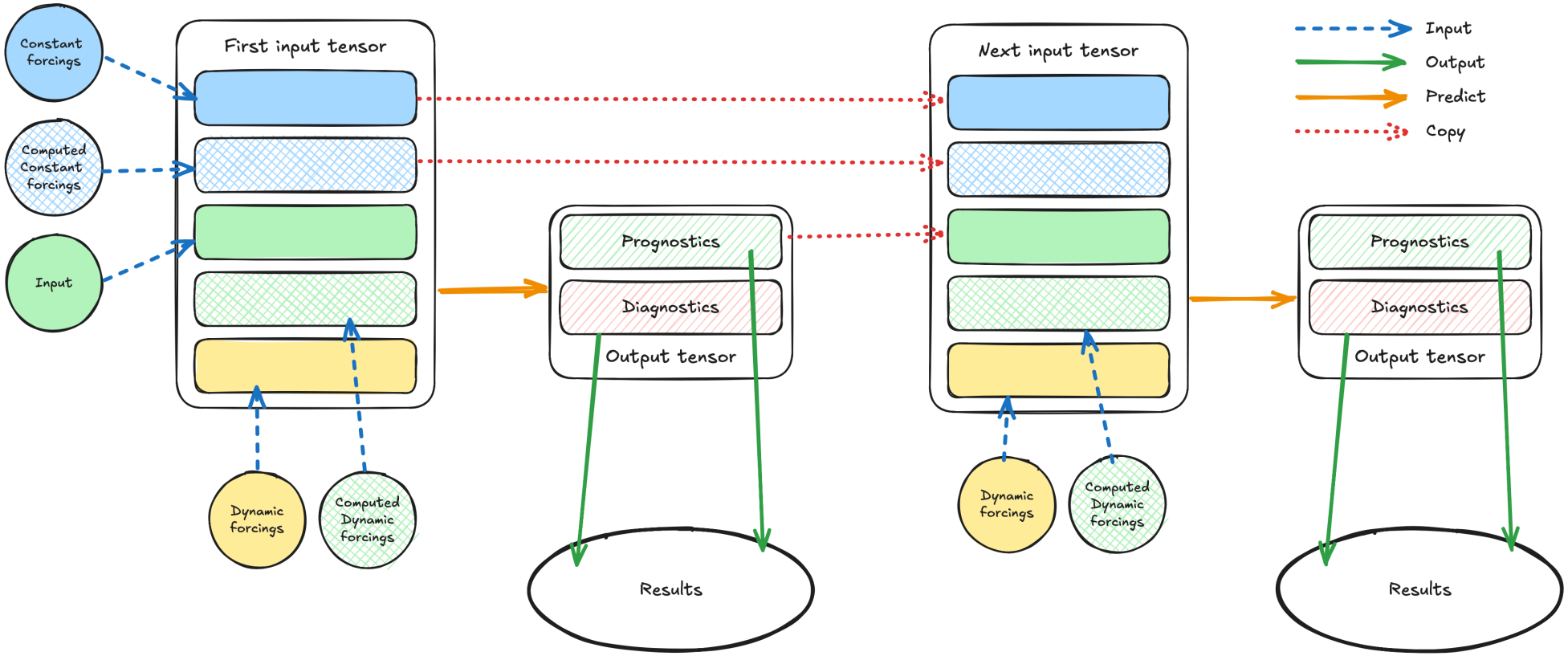


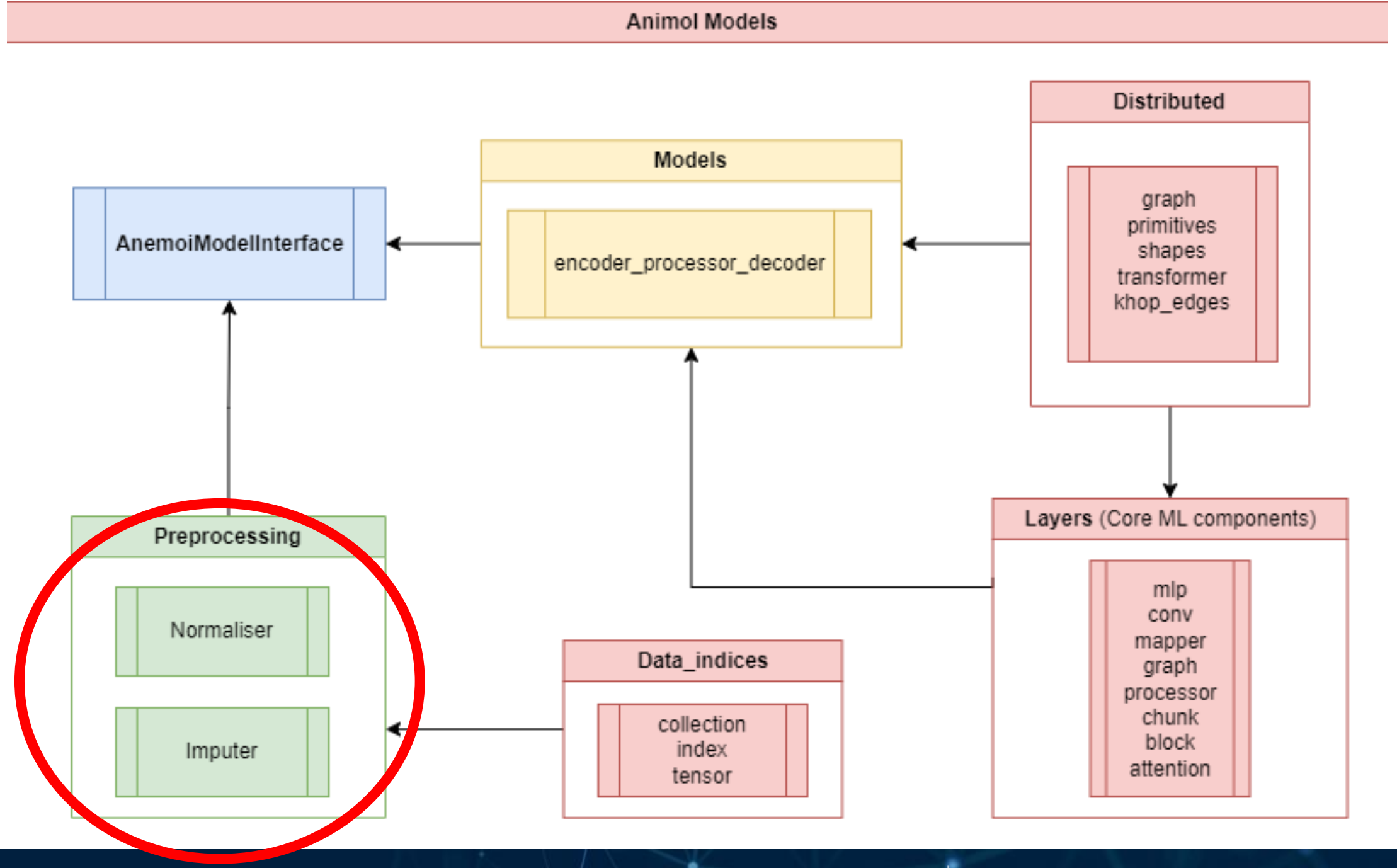

Animol Models
Distributed
graph
primitives
shapes
transformer
khop_edges
Models
encoder_processor_decoder
AnemoiModelInterface
Layers (Core ML components)
mlp
conv
mapper
graph
processor
chunk
block
attention
Preprocessing
Normaliser
Imputer
Data_indices
collection
index
tensor


Animol Models
Distributed
graph
primitives
shapes
transformer
khop_edges
Models
encoder_processor_decoder
AnemoiModelInterface
Layers (Core ML components)
mlp
conv
mapper
graph
processor
chunk
block
attention
Preprocessing
Normaliser
Imputer
Data_indices
collection
index
tensor

How to index the big array?

- Anemoi datasets, we built a big array.
- But we may use data differently.
- **Prognostic** (in and out), **forcing** (in), and **diagnostic** (out).
- Need code to understand how to get data in and out of the tensor.




Animol Models
Distributed
graph
primitives
shapes
transformer
khop_edges
Models
encoder_processor_decoder
AnemoiModelInterface
Layers (Core ML components)
mlp
conv
mapper
graph
processor
chunk
block
attention
Preprocessing
Normaliser
Imputer
Data_indices
collection
index
tensor

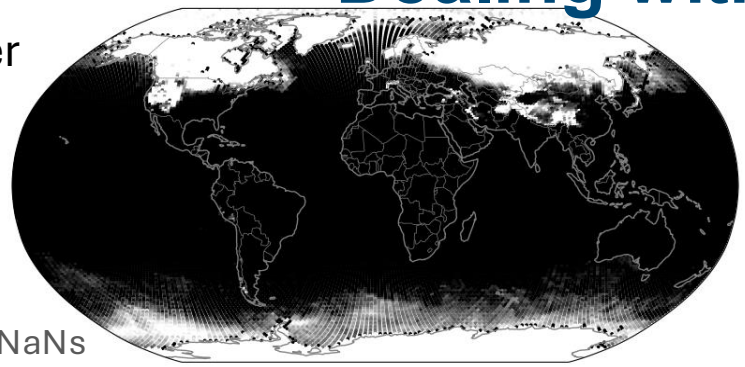
Normaliser

- Apply the statistics to normalise the data.
  - Could be in the training code.
  - But it living in the model helps with inference.
- Options:
  - Mean-std
  - Std
  - Max
  - Min-max
  - None

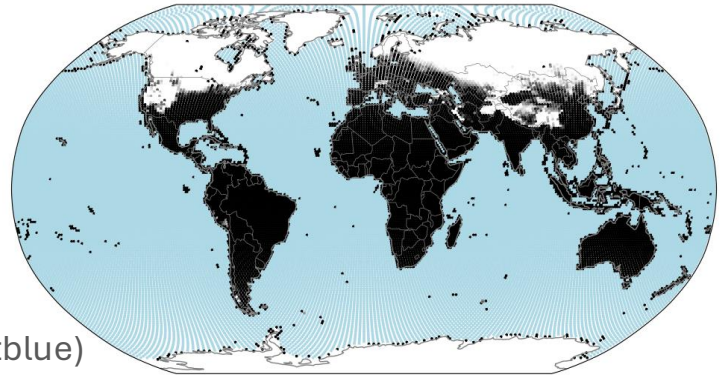
```yaml
normalizer:
  default: "mean-std"

  # Remap cp statistics to those of tp when using FractionBounding. This ensures
  # that cp, as a fraction of tp, remains consistent with tp's scale and statistics.
  # NOTE: This remap should only be applied if FractionBounding is enabled for cp.
  # remap:
  #   cp: tp

  # Standardization applied to tp and cp variables. Ensure that if cp is bounded
  # as a fraction of tp, both variables are normalized using these shared statistics.
  # "Std" normalization is preferred here over "mean-std" to avoid shifting of the
  # zero value in the normalized space.
  std:
  - "tp"
  # - "cp"

  min-max:
  max:
  - "sdor"
  - "slor"
  - "z"
  none:
  - "cos_latitude"
  - "cos_longitude"
  - "sin_latitude"
  - "sin_longitude"
  - "cos_julian_day"
  - "cos_local_time"
  - "sin_julian_day"
  - "sin_local_time"
  - "insolation"
  - "lsm"
```

Dealing with missing values

Snow Cover over ocean



Without masking NaNs



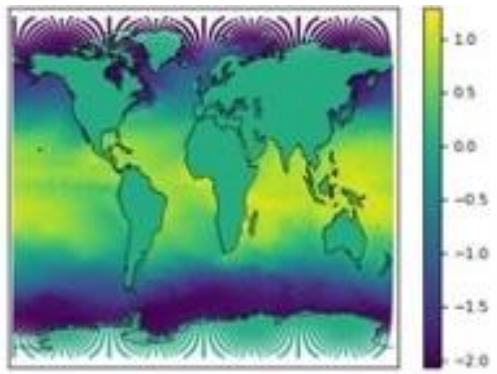
Masking NaNs (lightblue)

In training: NaNs are replaced by values and **masked in the loss function with zeros**

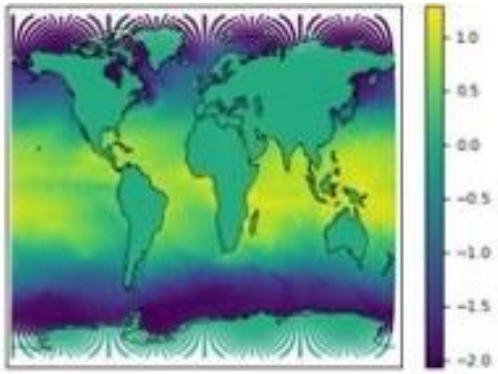
At inference: the model **'predicts'** values **in the masked areas** – these are **removed** in a **post-processing** step

Sea Surface Temperature over land

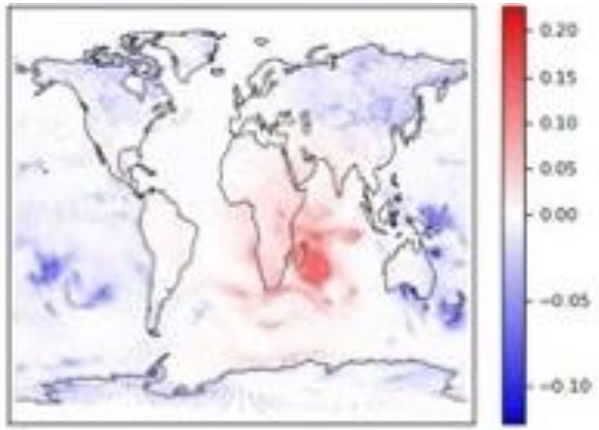
Target



Predicted field

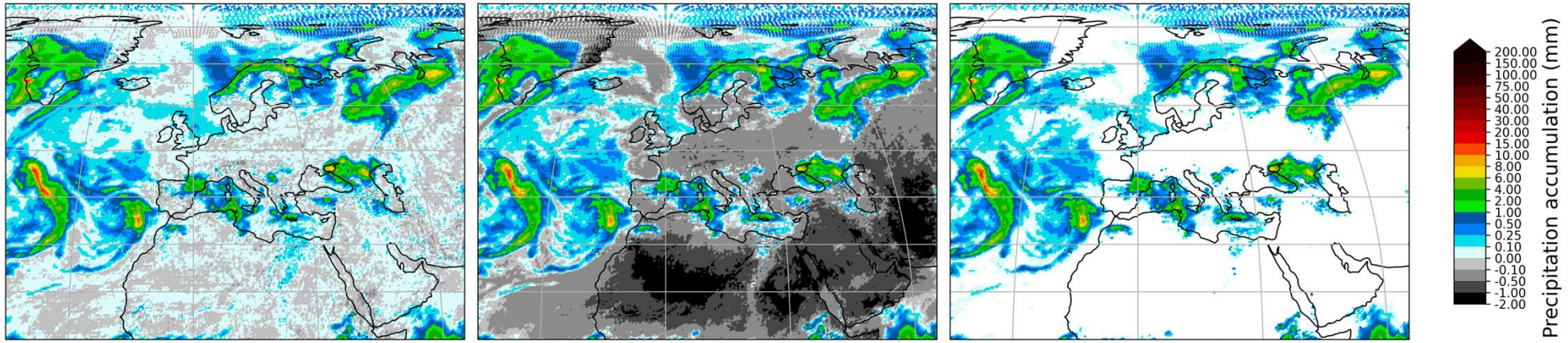


Predicted Increment



- *Additional issues caused when NaNs move! This was the case with waves fields.*

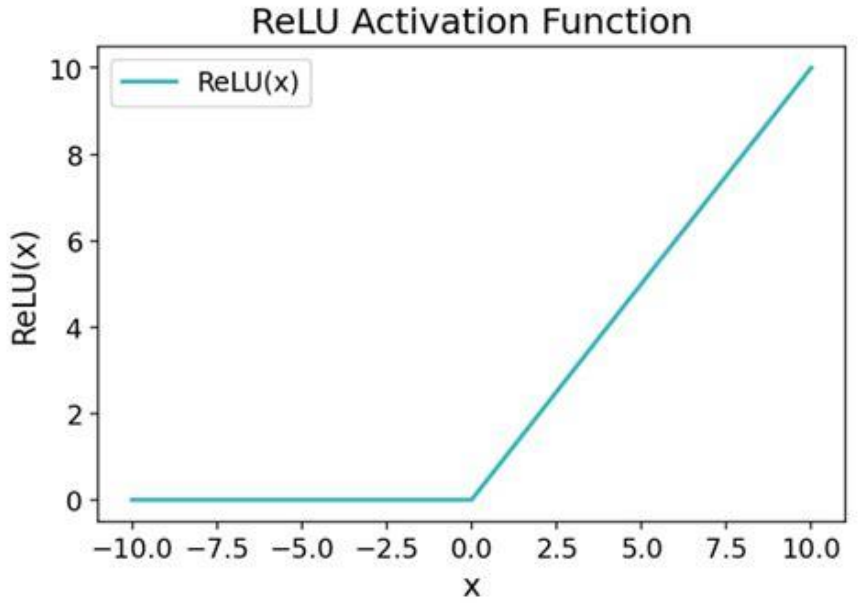
Bounding – a missing piece of the model diagram!



Previous AIFS

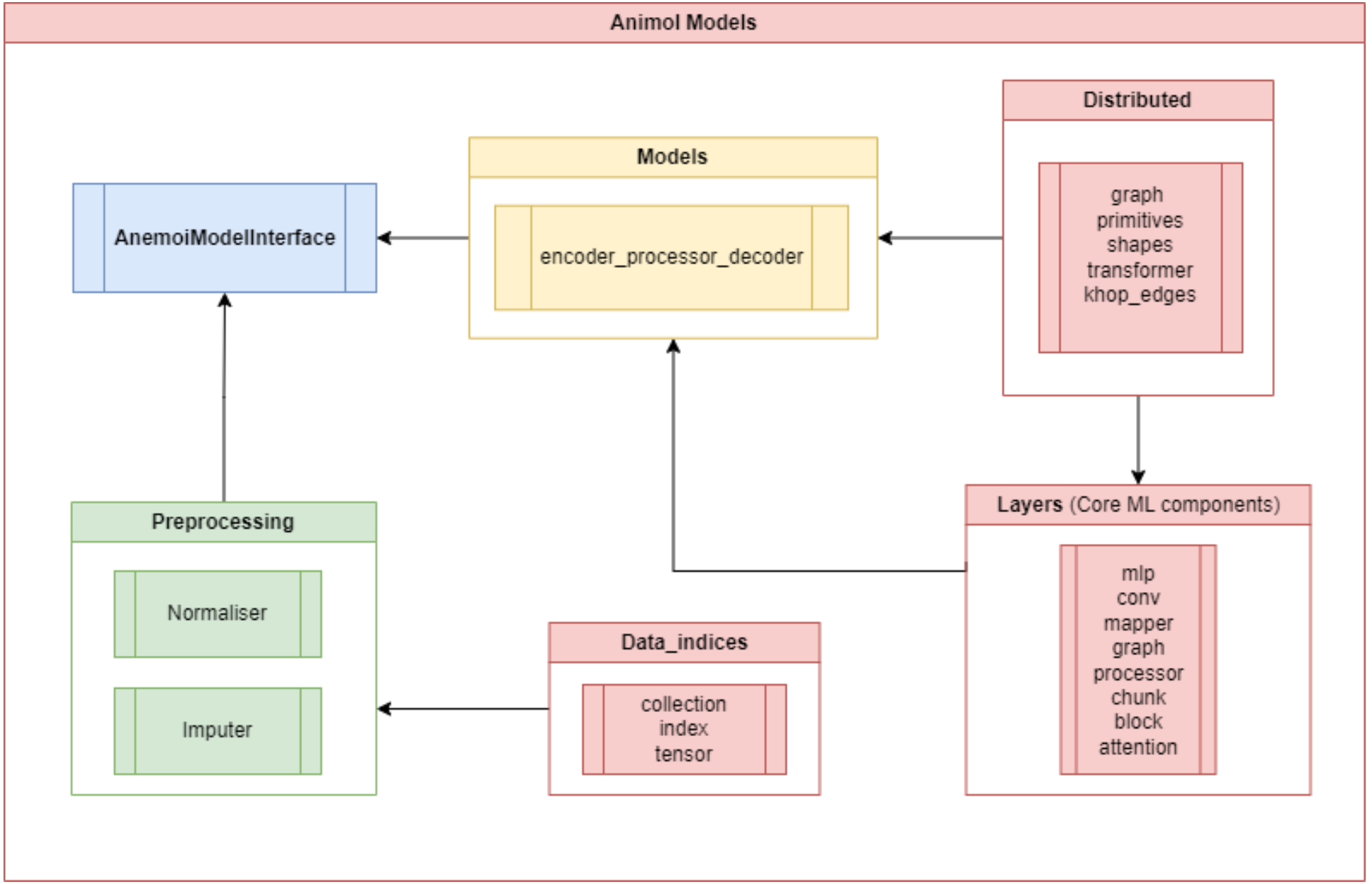
Input to bounding layer

New AIFS




```
# Bounding configuration
bounding: #These are applied in order

  # Bound tp (total precipitation) with a Relu bounding layer
  # ensuring a range of [0, infinity) to avoid negative precipitation values.
  - _target_: anemoi.models.layers.bounding.ReluBounding #[0, infinity)
    variables:
    - tp

  # [OPTIONAL] Bound cp (convective precipitation) as a fraction of tp.
  # This guarantees that cp is physically consistent with tp by restricting cp
  # to a fraction of tp [0 to 1]. Uncomment the lines below to apply.
  # NOTE: If this bounding strategy is used, the normalization of cp must be
  # changed to "std" normalization, and the "cp" statistics should be remapped
  # to those of tp to ensure consistency.

  # - _target_: anemoi.models.layers.bounding.FractionBounding # fraction of tp
  #   variables:
  #   - cp
  #   min_val: 0
  #   max_val: 1
  #   total_var: tp
```

Questions?


Animol Models
Distributed
graph
primitives
shapes
transformer
khop_edges
Models
encoder_processor_decoder
AnemoiModelInterface
Layers (Core ML components)
mlp
conv
mapper
graph
processor
chunk
block
attention
Preprocessing
Normaliser
Imputer
Data_indices
collection
index
tensor

Thank you!

anemoi-models.readthedocs.io

mario.santacruz@ecmwf.int

www.ecmwf.int

See you tomorrow at 15:00 CEST
Webinar 3: Anemoi Training + Anemoi Inference

The Teams link will be shared via email

www.ecmwf.int

training@ecmwf.int

https://events.ecmwf.int/event/562/