AtmoRep

Large scale representation learning of atmospheric dynamics

Christian Lessig, Ilaria Luise, Martin Schultz
Representation learning

- Learn a domain-specific but task-independent neural network that is useful for a range of applications
Representation learning

- Learn a domain-specific but task-independent neural network that is useful for a range of applications
  - Representation network provides transformation of network input to effective feature spaces
  - Self-supervised training on very large amounts of data (O (PB)) with very large networks (O (10^9) parameters)
  - Useful for downstream applications using tail network or fine-tuning
Can we perform representation learning in the Earth sciences?
Representation learning for the Earth sciences?

- Very large amounts of observational data
  - ERA5 reanalysis: 6+ PB
  - ESA’s MetO p-SG satellites: 8 x 864 GB/day
  - Data essentially completely unlabelled
Representation learning for the Earth sciences?

- Very large amounts of observational data
  - ERA5 reanalysis: 6+ PB
  - ESA’s MetOp-SG satellites: 8 x 864 GB/day
  - Data essentially completely unlabelled
Representation learning for the Earth sciences?

- Very large amounts of observational data
- No complete classical model for system and dynamics
  - Central issue for forecasting and climate projections
Representation learning for the Earth sciences?

- Very large amounts of observational data
- No complete classical model for system and dynamics
- Chaoticisty in atmospheric dynamics leads to ambiguity
  - There is often not one “correct answer”
  - Large networks learn statistical representations
AtmoRep

Large scale representation learning of atmospheric dynamics
AtmoRep

Historical observations

1950 1970 1990 2010
AtmoRep

Historical observations

ERA5 reanalysis

large scale machine learning

1950 1970 1990 2010
AtmoRep

Historical observations

1950 1970 1990 2010

ERA5 reanalysis

large scale machine learning

Forecasting

Impact analysis

Climate projections

Downscaling

AtmoRep
AtmoRep

- Historical observations
- ERA5 reanalysis
- Large scale machine learning

Applications:
- Forecasting
- Impact analysis
- Climate projections
- Downscaling
AtmoRep

Historical observations

ERA5 reanalysis

large scale machine learning

Forecasting
Impact analysis
Climate projections
Downscaling

scientific insight

applications
AtmoRep

Historical observations

ERA5 reanalysis

large scale
machine learning

Forecasting
Impact analysis
Climate projections
Downscaling

scientific insight

AtmoRep

applications
AtmoRep: longer term objectives
AtmoRep: longer term objectives

- Weather forecasting
  - Representation network as holistic closure for unresolved effects/basis for fully network based model
  - Address in particular challenging objectives such as precipitation, extreme events, sub-seasonal to seasonal
AtmoRep: longer term objectives

- Weather forecasting
- Climate projections
  - Representation network as holistic closure/parametrization for unresolved effects in a hybrid model
  - Ergodic hypothesis: use large spatial diversity (with local model) as substitute for lack of historical data
  - Stochastic climate models:\(^1\) representation network learns fast, stochastic scale of overall dynamics

AtmoRep: longer term objectives

- Weather forecasting
- Climate projections
- Coupled Earth system
  - Atmosphere, ocean, infrastructure, ...
AtmoRep: longer term objectives

- Weather forecasting
- Climate projections
- Coupled Earth system
- Scientific model
  - Can we use a large representation network as a scientific model complementing existing ones such as partial differential equations?
AtmoRep: longer term objectives

- Weather forecasting
- Climate projections
- Coupled Earth system
- Scientific model
- Training/fine-tuning on direct observational data
  - Learn directly from satellite observations and other measurements
Data: ERA5 reanalysis
Data: ERA5 reanalysis
Data: ERA5 reanalysis
Data: ERA5 reanalysis
Data: ERA5 reanalysis

137 vertical layers
Data: ERA5 reanalysis

721x1440 horizontal grid (0.25 degree)

137 vertical layers
Data: ERA5 reanalysis

- vorticity
- divergence
- temperature
- geopotential
- ...

137 vertical layers

721x1440 horizontal grid (0.25 degree)
Data: ERA5 reanalysis

- 721x1440 horizontal grid (0.25 degree)
- 137 vertical layers
- hourly for 70 years
- vorticity
- divergence
- temperature
- geopotential
- ...
Data: ERA5 reanalysis

- vorticity
- divergence
- temperature
- geopotential
- ...

721x1440 horizontal grid (0.25 degree)

137 vertical layers

over 6 PB of data readily amenable to machine learning

hourly for 70 years
AtmoRep data

temperature
AtmoRep data
AtmoRep data

vorticity
AtmoRep data

divergence
AtmoRep data

- Vorticity
- Temperature
- Divergence
- Geopotential
AtmoRep data
AtmoRep data
ERA5 versus ImageNet

![Graph showing comparison between ERA5 and ImageNet vorticity](image-url)
ERA5 versus ImageNet

- stream function
- velocity potential
- velocity
- vector field
- vorticity
- divergence
ERA5 versus ImageNet
ERA5 versus ImageNet

\[ \approx \]

Stream fct
ERA5 versus ImageNet

- stream function
- velocity potential
- velocity
- vector field
- vorticity
- divergence
AtmoRep network architecture
AtmoRep network architecture

- Transformer-based as network architecture
  - Scales well to very large data-sets
  - Generative model (with decoder)
  - Attention maps provide (physical) interpretability
AtmoRep network architecture

- Transformer-based as network architecture
  - Scales well to very large data-sets
  - Generative model (with decoder)
  - Attention maps provide (physical) interpretability
- Network is local in space-time
AtmoRep network architecture
AtmoRep network architecture
AtmoRep network architecture

- Transformer encoder-based network architecture
  - Scales well to very large data-sets
  - Generative model (with decoder)
  - Attention maps provide (physical) interpretability
- Network is local in space-time
  - Physics of dynamics are universally valid
  - Local particularities can be learned by providing time + space position as auxiliary information
What is a token?
What is a token?

token: small neighborhood in space-time
What is a token?
What is a token?

- Token is small neighborhood in space-time
  - Small for token attention / interaction to be informative
  - Big enough so token has rich internal structure
What is a token?

- Token is small neighborhood in space-time
  - Small for token attention / interaction to be informative
  - Big enough so token has rich internal structure
- Token size is field-dependent
What is a token?

- Token is small neighborhood in space-time
  - Small for token attention / interaction to be informative
  - Big enough so token has rich internal structure
- Token size is field-dependent
- Multiple token sizes to provide multi-resolution structure and larger contexts
Multiformer
Self-attention

Self-Attention MLP Self-Attention MLP Self-Attention MLP
Self attention

vorticity

Self-Attention  MLP  Self-Attention  MLP  ...  Self-Attention  MLP
Self attention

- Self-Attention
- MLP
- Self-Attention
- MLP
- Self-Attention
- MLP

divergence

vorticity
Multiformer

Self-attention

vorticity

divergence
temperature

geopotential

Self-Attention MLP Self-Attention MLP Self-Attention MLP

…

Self-Attention MLP Self-Attention MLP Self-Attention MLP
Self-Attention

MLP

geopotential

temperature

divergence

vorticity

Self-Attention

MLP

Self-Attention

MLP

Self-Attention

MLP

Self-Attention

MLP

Self-Attention

MLP
Self-attention

Cross attention

geopotential

temperature

divergence

vorticity

Self-Attention

MLP

Self-Attention

MLP

Self-Attention

MLP

Self-Attention

MLP
Self-attention

Cross attention

geopotential

temperature

divergence

vorticity

Self-Attention MLP

Self-Attention MLP

Self-Attention MLP

Multiformer
Multiformer

vorticity

divergence
Multiformer

- Different physical fields with different properties have separate latent spaces (and transformations for these)
- Individual fields can be pre-trained independently
- Plug-and-play of fields
  - Fields can be added/removed with limited (or no) computational effort
- Cross-attention allows for explicit introspection of interaction between fields
Embedding of tokens

multiformer
Embedding of tokens
Embedding of tokens

data loader

embed

multiformer

tail
Embedding network

- Multiformer models longer range effects and field interactions in a rich latent space
Embedding network

- Multiformer models longer range effects and field interactions in a rich latent space
  - Embedding network provides rich encoding of input field
Embedding network

- Multiformer models longer range effects and field interactions in a rich latent space
  - Embedding network provides rich encoding of input field
  - Embedding network allows for multi-resolution representation per field, i.e. different token sizes
Embedding of tokens
Embedding of tokens

- Multiformer models longer range effects and field interactions in a rich latent space
  - Embedding network provides rich encoding of input field
  - Embedding network allows for multi-resolution representation per field, i.e. different token sizes
Embedding of tokens

- Multiformer models longer range effects and field interactions in a rich latent space
  - Embedding network provides rich encoding of input field
  - Embedding network allows for multi-resolution representation per field, i.e. different token sizes

Use transformer as embedding network
Embedding of tokens
Embedding of tokens
Embedding of tokens
Embedding of tokens
Embedding of tokens
Embedding of tokens

Diagram showing the process of embedding tokens with CLS.
Training

- Unbiased hierarchical Monte Carlo sampling of all possible ERA5 space-time cubes
Training
Training
Training

1979 \( t \rightarrow 2020\)

03/1984 \( \rightarrow 09/2003\)
Training

© Christian Lessig, 2022

1979 \( t \) \[\rightarrow\] 03/1984 \[\rightarrow\] 09/2003 \[\rightarrow\] 2020
Training

area preserving sampling of sphere

1979 $t$  

03/1984 09/2003

2020
Training

area preserving sampling of sphere

1979 \[ t \] 

03/1984 09/2003

2020
Training

- Unbiased hierarchical Monte Carlo sampling of all possible ERA5 space-time cubes
  - Random sampling of (year,month) tuples corresponding to individual files
  - Random sampling of space-time cubes in tuples
  - Trivially parallelizable with one data loader per field
- Area preserving sampling for sphere/Earth to compensate for distortion of regular grid
Spatio-temporal BERT

- Self-supervised training with variation of BERT masked language (or token) model
Spatio-temporal BERT

Flatland view
Spatio-temporal BERT

Flatland view of BERT
Spatio-temporal BERT

- Self-supervised training with variation of BERT masked language language model
  - Natural interpretation as forecasting / hindcasting / interpolation
Spatio-temporal BERT

- Self-supervised training with variation of BERT masked language language model
  - Natural interpretation as forecasting / hindcasting / interpolation
  - Bi-directional training poses interesting physical questions
Spatio-temporal BERT

- Self-supervised training with variation of BERT masked language model
  - Natural interpretation as forecasting / hindcasting / interpolation
  - Bi-directional training poses interesting physical questions
  - Performed on randomly cropped subset to be invariant to specific token numbers
Spatio-temporal BERT
Spatio-temporal BERT
Spatio-temporal BERT
Statistical loss

- ML: Training on MSE loss is problematic in terms of training dynamics
Statistical loss

- ML: Training on MSE loss is problematic in terms of training dynamics
- Training on just the mean is sub-optimal to learn a probabilistic/statistical representation of the dynamics and the system
Statistical loss


Adams Wei Yu, David Dohan, Minh-Thang Luong, Rui Zhao, Kai Chen, Mohammad Norouzi, and Quoc V Le. 2018. QANet: Combining local convolution with global self-attention for reading comprehension. In ICLR.


Appendix for “BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding”

- 80% of the time: Replace the word with the [MASK] token, e.g., my dog is hairy → my dog is [MASK]

- 10% of the time: Replace the word with a random word, e.g., my dog is hairy → my dog is apple

- 10% of the time: Keep the word unchanged, e.g., my dog is hairy → my dog is hairy. The purpose of this is to bias the representation towards the actual observed word.

The advantage of this procedure is that the Transformer encoder does not know which words it will be asked to predict or which have been replaced by random words, so it is forced to keep a distributional contextual representation of every input token. Additionally, because random replacement only occurs for 1.5% of all tokens (i.e., 10% of 15%), this does not seem to harm the model’s language understanding capability. In Section C.2, we evaluate the impact this procedure.
Statistical loss

How to obtain improved training dynamics and ensure probabilistic/statistical representation in network?
Statistical loss
Statistical loss
Statistical loss
Statistical loss
Statistical loss
Statistical loss
Statistical loss
Statistical loss
Statistical loss
Statistical loss
Statistical loss
Statistical loss
Statistical loss

end-to-end training encourages statistical representation
Statistical loss: experiments

- BERT with conditional masking
- 975 hPa (high frequency) vorticity
- 40 years of training data
Statistical loss

MSE test loss

- no ensemble, MSE
Statistical loss

MSE test loss

- no ensemble, MSE
- ensemble=10, MSE+stats
Statistical loss

- Statistical loss:
Statistical loss

- Statistical loss:

- CRPS:\(^1\)

---

Statistical loss

\[ \text{ensemble} = 10, \text{MSE+CRPS} \]

\[ \text{ensemble} = 10, \text{MSE+stats} \]
Statistical loss

- Predictions:

[Images of heatmaps and line graphs showing prediction and reference data]
Statistical loss

- Predictions:
Statistical loss

- Predictions:

![Prediction](image1)

![Reference](image2)

![Graph](image3)
Statistical loss

Histogram of ensemble errors
Statistical loss

2D Histogram of $L_2$ error vs. std. dev.
Statistical loss

- Attention maps:

- **Attention maps**

- **Vorticity**

- **t-2**

- **t-1**

- **t**
Zero shot evaluation

- Evaluate performance on representation network as is
Zero shot evaluation

- Evaluate performance on representation network as is

Proposal for atmospheric representation learning:
Short term now-/forecasting
Zero shot evaluation
Zero shot evaluation

© Christian Lessig, 2022
Zero shot evaluation
Zero shot evaluation
Zero shot evaluation
Zero shot evaluation
Zero shot evaluation

Embedding-Forecast

future

past
Zero shot evaluation
Zero shot evaluation
Zero shot evaluation
Zero shot performance

MSE

persistence
Zero shot performance

MSE

in training

persistence
Zero shot performance

MSE

- in training
- zero shot, embedding
- persistence
Zero shot performance

MSE

in training
zero shot, BERT
zero shot, embedding
persistence
Zero shot performance

**MSE**

- 0.35, conditional
- 0.25, conditional
- 0.15, conditional
- 0.15, no ensemble
- 0.075, conditional
- 1 token, no conditional
Zero shot performance

- BERT
  - 0.35, conditional
  - 0.25, conditional
  - 0.15, conditional
  - 0.15, no ensemble
  - 0.075, conditional
  - 1 token, no conditional

Final test loss
Zero shot performance

BERT (0.15) per training data

- 40 years
- 10 years
- 5 years
- 2 years
Zero shot performance

BERT (0.15) per training data

MSE

- 40 years
- 10 years
- 5 years
- 2 years

final test loss
AtmoRep: longer term objectives

- Weather forecasting
- Climate projections
- Coupled Earth system
- Scientific model
- Training/fine-tuning on direct observational data
3.1 Encoder and Decoder Stacks

**Encoder:**
The encoder is composed of a stack of \( N = 6 \) identical layers. Each layer has two sub-layers. The first is a multi-head self-attention mechanism, and the second is a simple, position-wise fully connected feed-forward network. We employ a residual connection \([11]\) around each of the two sub-layers, followed by layer normalization \([1]\). That is, the output of each sub-layer is \( \text{LayerNorm}(x + \text{Sublayer}(x)) \), where \( \text{Sublayer}(x) \) is the function implemented by the sub-layer itself. To facilitate these residual connections, all sub-layers in the model, as well as the embedding layers, produce outputs of dimension \( d_{\text{model}} = 512 \).

**Decoder:**
The decoder is also composed of a stack of \( N = 6 \) identical layers. In addition to the two sub-layers in each encoder layer, the decoder inserts a third sub-layer, which performs multi-head attention over the output of the encoder stack. Similar to the encoder, we employ residual connections around each of the sub-layers, followed by layer normalization. We also modify the self-attention sub-layer in the decoder stack to prevent positions from attending to subsequent positions. This masking, combined with fact that the output embeddings are offset by one position, ensures that the predictions for position \( i \) can depend only on the known outputs at positions less than \( i \).

3.2 Attention

An attention function can be described as mapping a query and a set of key-value pairs to an output, where the query, keys, values, and output are all vectors. The output is computed as a weighted sum of the values, where the weight assigned to each value is computed by a compatibility function of the query with the corresponding key.

**Forecasting / projections**

The Transformer - model architecture.

### 3.1 Encoder and Decoder Stacks

**Encoder:** The encoder is composed of a stack of \( N = 6 \) identical layers. Each layer has two sub-layers. The first is a multi-head self-attention mechanism, and the second is a simple, position-wise fully connected feed-forward network. We employ a residual connection [11] around each of the two sub-layers, followed by layer normalization [1]. That is, the output of each sub-layer is

\[
\text{LayerNorm}(x + \text{Sublayer}(x))
\]

where \( \text{Sublayer}(x) \) is the function implemented by the sub-layer itself. To facilitate these residual connections, all sub-layers in the model, as well as the embedding layers, produce outputs of dimension \( d_{\text{model}} = 512 \).

**Decoder:** The decoder is also composed of a stack of \( N = 6 \) identical layers. In addition to the two sub-layers in each encoder layer, the decoder inserts a third sub-layer, which performs multi-head attention over the output of the encoder stack. Similar to the encoder, we employ residual connections around each of the sub-layers, followed by layer normalization. We also modify the self-attention sub-layer in the decoder stack to prevent positions from attending to subsequent positions. This masking, combined with the fact that the output embeddings are offset by one position, ensures that the predictions for position \( i \) can depend only on the known outputs at positions less than \( i \).

### 3.2 Attention

An attention function can be described as mapping a query and a set of key-value pairs to an output, where the query, keys, values, and output are all vectors. The output is computed as a weighted sum of the values, where the weight assigned to each value is computed by a compatibility function of the query with the corresponding key.

Autoregressive, generative modeling

3.1 Encoder and Decoder Stacks

Encoder:
The encoder is composed of a stack of $N = 6$ identical layers. Each layer has two sub-layers. The first is a multi-head self-attention mechanism, and the second is a simple, position-wise fully connected feed-forward network. We employ a residual connection [11] around each of the two sub-layers, followed by layer normalization [1]. That is, the output of each sub-layer is $\text{LayerNorm}(x + \text{Sublayer}(x))$, where $\text{Sublayer}(x)$ is the function implemented by the sub-layer itself. To facilitate these residual connections, all sub-layers in the model, as well as the embedding layers, produce outputs of dimension $d_{\text{model}} = 512$.

Decoder:
The decoder is also composed of a stack of $N = 6$ identical layers. In addition to the two sub-layers in each encoder layer, the decoder inserts a third sub-layer, which performs multi-head attention over the output of the encoder stack. Similar to the encoder, we employ residual connections around each of the sub-layers, followed by layer normalization. We also modify the self-attention sub-layer in the decoder stack to prevent positions from attending to subsequent positions. This masking, combined with the fact that the output embeddings are offset by one position, ensures that the predictions for position $i$ can depend only on the known outputs at positions less than $i$.

3.2 Attention

An attention function can be described as mapping a query and a set of key-value pairs to an output, where the query, keys, values, and output are all vectors. The output is computed as a weighted sum of the values, where the weight assigned to each value is computed by a compatibility function of the query with the corresponding key.


Forecasting / projections

autoregressive, generative modeling

akin to time stepping loop (roll out) for forecasting/projections
3.1 Encoder and Decoder Stacks

**Encoder:** The encoder is composed of a stack of \( N = 6 \) identical layers. Each layer has two sub-layers. The first is a multi-head self-attention mechanism, and the second is a simple, position-wise fully connected feed-forward network. We employ a residual connection \([11]\) around each of the two sub-layers, followed by layer normalization \([1]\). That is, the output of each sub-layer is
\[
\text{LayerNorm}(x + \text{Sublayer}(x)),
\]
where \( \text{Sublayer}(x) \) is the function implemented by the sub-layer itself. To facilitate these residual connections, all sub-layers in the model, as well as the embedding layers, produce outputs of dimension \( d_{\text{model}} = 512 \).

**Decoder:** The decoder is also composed of a stack of \( N = 6 \) identical layers. In addition to the two sub-layers in each encoder layer, the decoder inserts a third sub-layer, which performs multi-head attention over the output of the encoder stack. Similar to the encoder, we employ residual connections around each of the sub-layers, followed by layer normalization. We also modify the self-attention sub-layer in the decoder stack to prevent positions from attending to subsequent positions. This masking, combined with the fact that the output embeddings are offset by one position, ensures that the predictions for position \( i \) can depend only on the known outputs at positions less than \( i \).

3.2 Attention

An attention function can be described as mapping a query and a set of key-value pairs to an output, where the query, keys, values, and output are all vectors. The output is computed as a weighted sum of the values, where the weight assigned to each value is computed by a compatibility function of the query with the corresponding key.

3.1 Encoder and Decoder Stacks

Encoder: The encoder is composed of a stack of \( N = 6 \) identical layers. Each layer has two sub-layers. The first is a multi-head self-attention mechanism, and the second is a simple, position-wise fully connected feed-forward network. We employ a residual connection \([\text{LayerNorm}(x + \text{Sublayer}(x))] \) around each of the two sub-layers, followed by layer normalization. That is, the output of each sub-layer is computed as:

\[
\text{LayerNorm}(x + \text{Sublayer}(x))
\]

where \( \text{Sublayer}(x) \) is the function implemented by the sub-layer itself. To facilitate these residual connections, all sub-layers in the model, as well as the embedding layers, produce outputs of dimension \( d_{\text{model}} = 512 \).

Decoder: The decoder is also composed of a stack of \( N = 6 \) identical layers. In addition to the two sub-layers in each encoder layer, the decoder inserts a third sub-layer, which performs multi-head attention over the output of the encoder stack. Similar to the encoder, we employ residual connections around each of the sub-layers, followed by layer normalization. We also modify the self-attention sub-layer in the decoder stack to prevent positions from attending to subsequent positions. This masking, combined with the fact that the output embeddings are offset by one position, ensures that the predictions for position \( i \) can depend only on the known outputs at positions less than \( i \).

3.2 Attention

An attention function can be described as mapping a query and a set of key-value pairs to an output, where the query, keys, values, and output are all vectors. The output is computed as a weighted sum of the values, where the weight assigned to each value is computed by a compatibility function of the query with the corresponding key.

Forecasting / projections

- Local model: apply model locally but couple through loss to get consistent forecast\(^1\)
  - Memory efficient and easily parallelizable

Forecasting / projections

- Local model: apply model locally but couple through loss to get consistent forecast\(^1\)
  - Memory efficient and easily parallelizable
- How to represent multi-res structure in a transformer?
  - Number of tokens or embedding dimension?
  - Couple it in UNet like structure between encoder and decoder

Current/next steps

- Complete representation learning model
  - Scale data and network size
  - Different training tasks and protocols
Current/next steps

- Complete representation learning model
- Downstream applications
  - Weather forecasting
  - Hurricane tracking
  - Wind farm placement
  - Air pollution forecasting
  - ...
Current/next steps

- Complete representation learning model
- Downstream applications
- Strong convection events
  - Improve forecasting
  - Study representation of physics in the network
AtmoRep

Large scale representation learning of atmospheric dynamics
AtmoDist: evaluation

![Graph showing the time difference between samples and average loss. The graph compares 'mse' and 'ours' methods. The average loss increases over time, with 'ours' generally performing better than 'mse'.](image-url)