SPATIAL ECOLOGY

International Institute for Applied Systems Analysis
IIASA

Foundation Models for Geospatial Data

Antonio Fonseca

GeoComp & ML (2023)

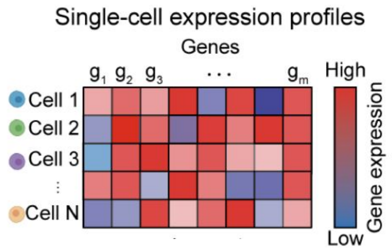
GeoComp & ML (2024)

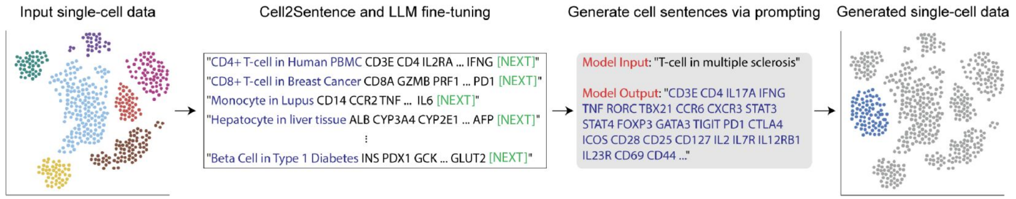
Agenda

- Intro:
    - Frameworks of learning
- The building block of modern ML
    - Transformers
    - Applications: language and image
- Foundation Models (FM)
    - FM vs conventional ML models
    - Prithvi: geospatial foundational model (IBM-Nasa)
    - Applications
- Tutorial

About me

- PhD in Neuroscience
- Foundation models for biomedical application





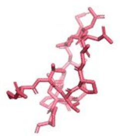
Input single-cell data
Cell2Sentence and LLM fine-tuning
Generate cell sentences via prompting
Generated single-cell data

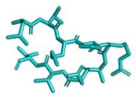
"CD4+ T-cell in Human PBMC CD3E CD4 IL2RA ... IFNG [NEXT]"
"CD8+ T-cell in Breast Cancer CD8A GZMB PRF1 ... PD1 [NEXT]"
"Monocyte in Lupus CD14 CCR2 TNF ... IL6 [NEXT]"
"Hepatocyte in liver tissue ALB CYP3A4 CYP2E1 ... AFP [NEXT]"
⋮
"Beta Cell in Type 1 Diabetes INS PDX1 GCK ... GLUT2 [NEXT]"

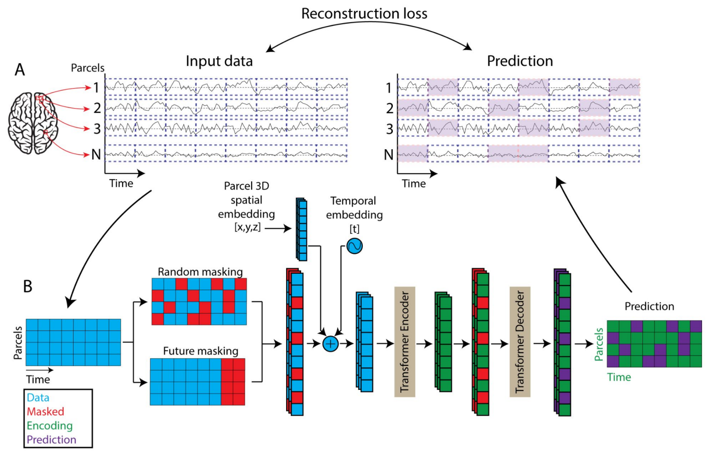
Model Input: "T-cell in multiple sclerosis"

Model Output: "CD3E CD4 IL17A IFNG TNF RORC TBX21 CCR6 CXCR3 STAT3 STAT4 FOXP3 GATA3 TIGIT PD1 CTLA4 ICOS CD28 CD25 CD127 IL2 IL7R IL12RB1 IL23R CD69 CD44 ..."

Drug discovery

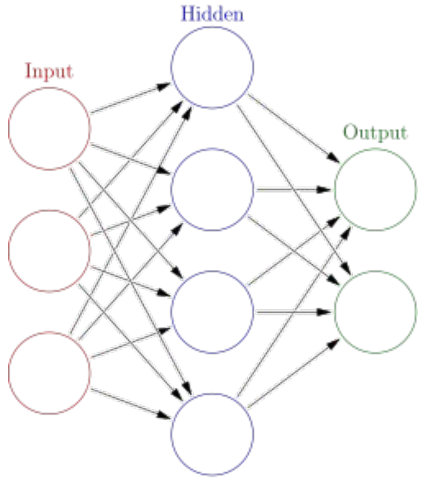






Brain recordings

From supervised learning to Foundation Models

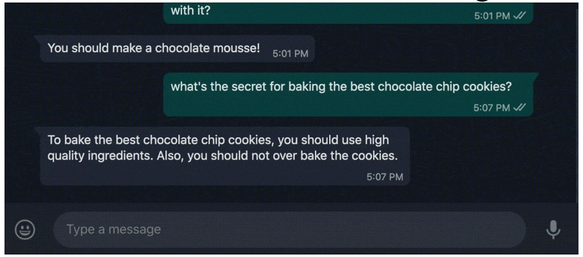


Neural Net for Supervised Learning

task-specific ML models

???

ChatGPT: conversational agent



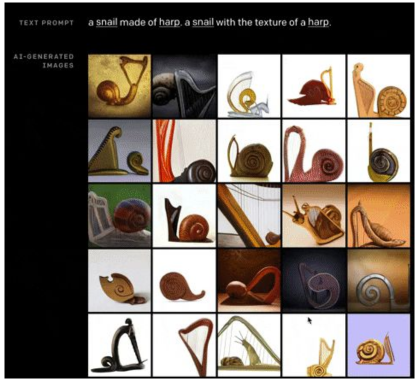
with it? 5:01 PM

You should make a chocolate mousse! 5:01 PM

what's the secret for baking the best chocolate chip cookies? 5:07 PM

To bake the best chocolate chip cookies, you should use high quality ingredients. Also, you should not over bake the cookies. 5:07 PM

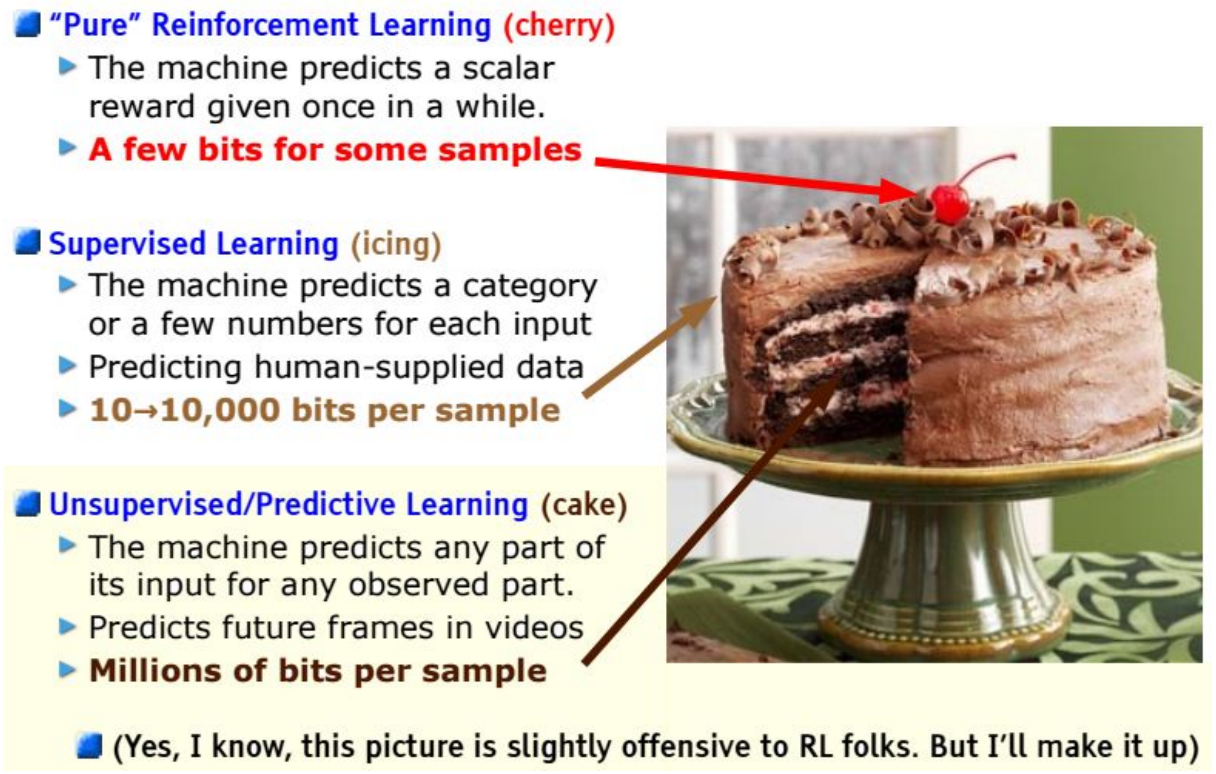
Dall-E: generate image from prompt



general-purpose ML models

Frameworks for Machine Learning

Most real-world problems fall into the category of unsupervised learning.



- "Pure" Reinforcement Learning (cherry)
  - The machine predicts a scalar reward given once in a while.
  - **A few bits for some samples**
- Supervised Learning (icing)
  - The machine predicts a category or a few numbers for each input
  - Predicting human-supplied data
  - **10→10,000 bits per sample**
- Unsupervised/Predictive Learning (cake)
  - The machine predicts any part of its input for any observed part.
  - Predicts future frames in videos
  - **Millions of bits per sample**
- (Yes, I know, this picture is slightly offensive to RL folks. But I'll make it up)

Original LeCun cake analogy slide presented at NIPS 2016

Unsupervised Learning techniques

1. **Clustering**:
   - Grouping similar data points together.
   - Example: K-means, Hierarchical Clustering, Spectral Clustering
2. **Dimensionality Reduction**:
   - Reducing the number of features.
   - Example: PCA, UMAP, VAE.
3. **Sequence learning**:
   - Learning patterns in series
   - Example 1: RNNs, LSTMs
   - Example 2: BERT, GPT, LLaMA (all based on Transformers).

Transformers *vs* Neural Networks

What Separates Transformers from Previous Neural Networks?

Self-Attention Mechanism:

- Allows the model to focus on different parts of the input sequence when making predictions.
- Captures long-range dependencies more effectively than traditional RNNs (Recurrent Neural Networks).

Parallel Processing:

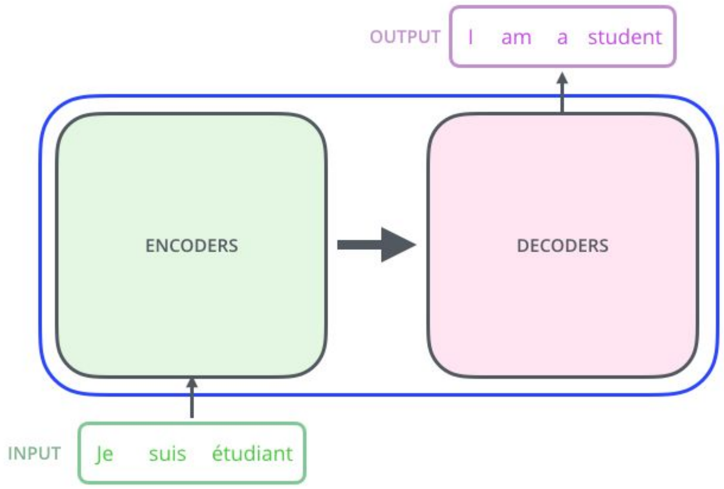
- Transformers process all tokens in the input sequence simultaneously.
- Faster training times compared to sequential processing in RNNs and LSTMs (Long Short-Term Memory networks).

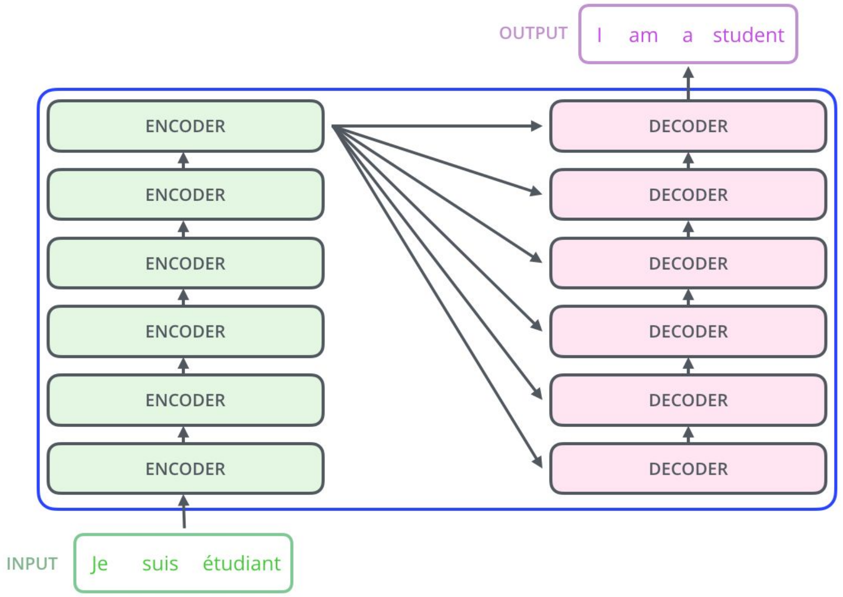
Versatility:

- Applied to various tasks beyond NLP, including image processing and multimodal learning.
- Unified architecture for diverse applications.

How are Transformers implemented ?

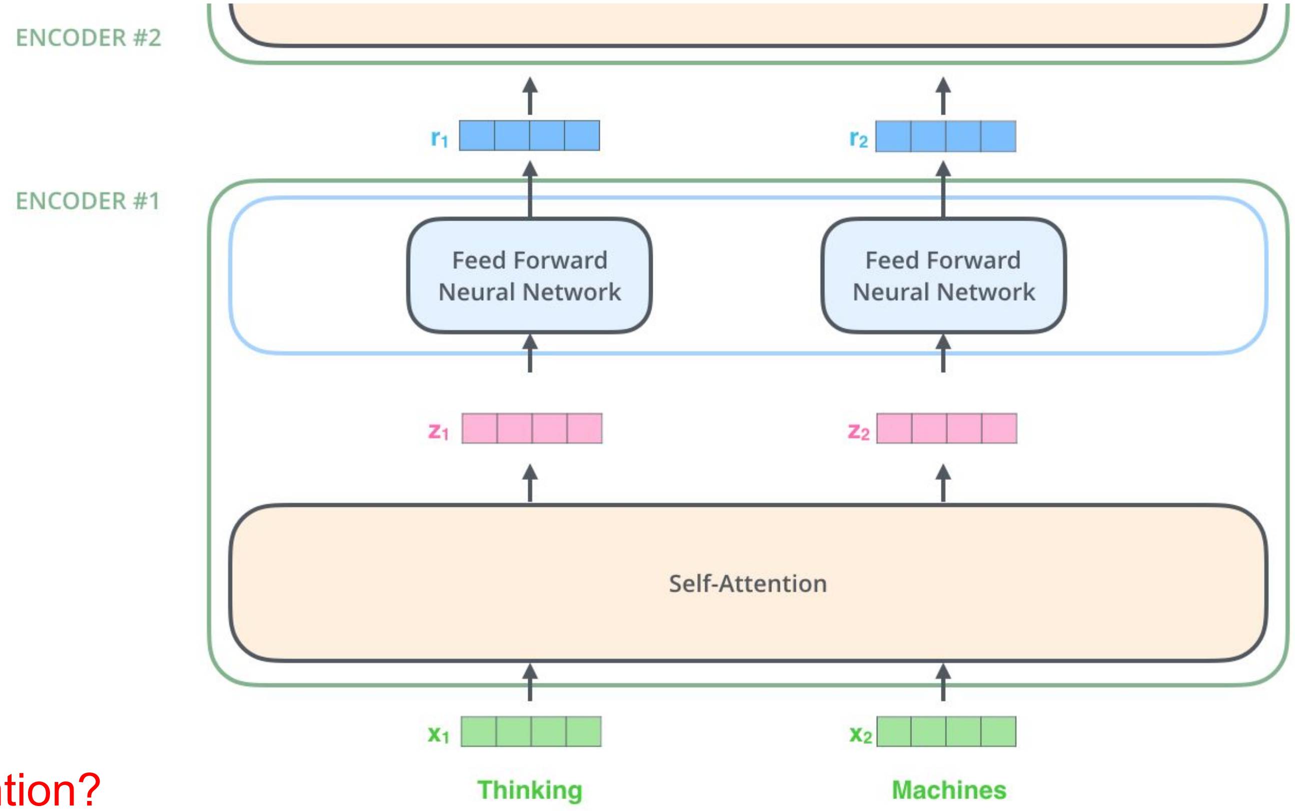
Transformer in high-level





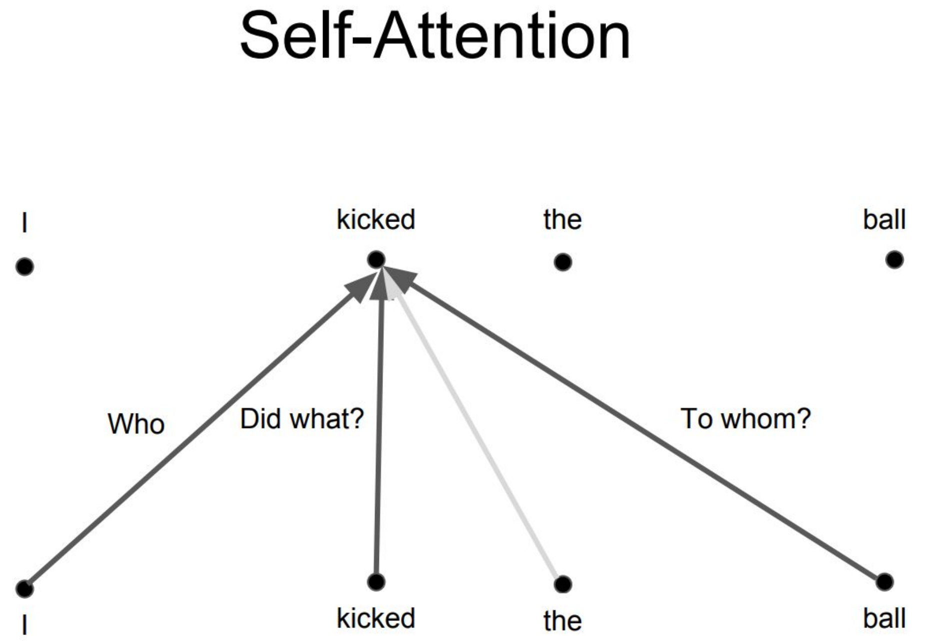
What is inside the blocks?

Each Encoder has Self-Attention



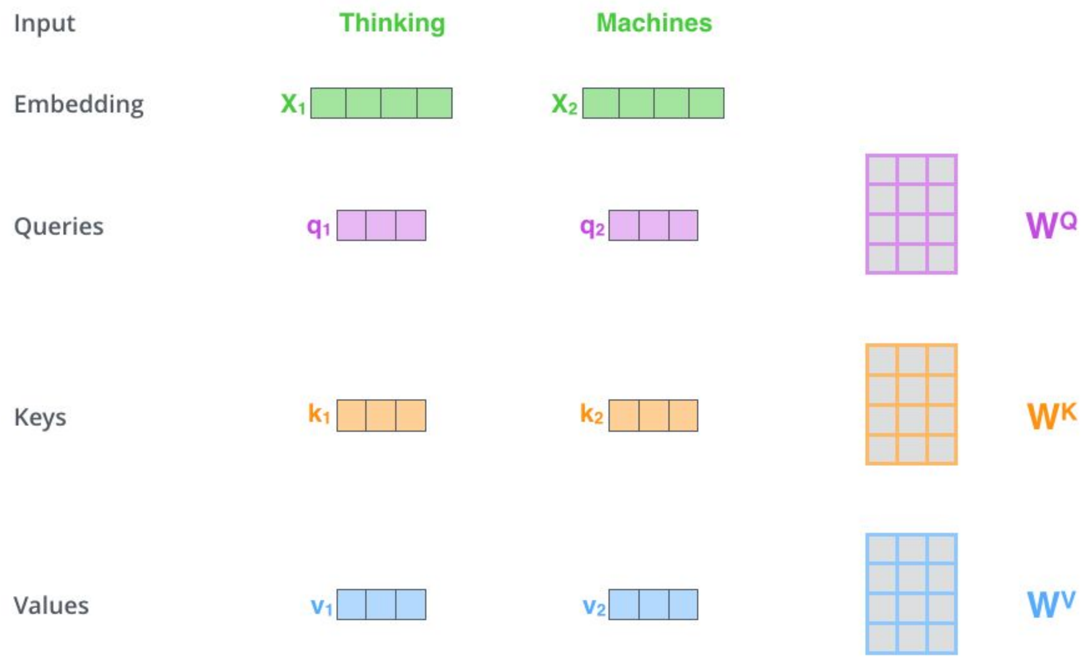
What is attention?

Self-Attention at a High Level



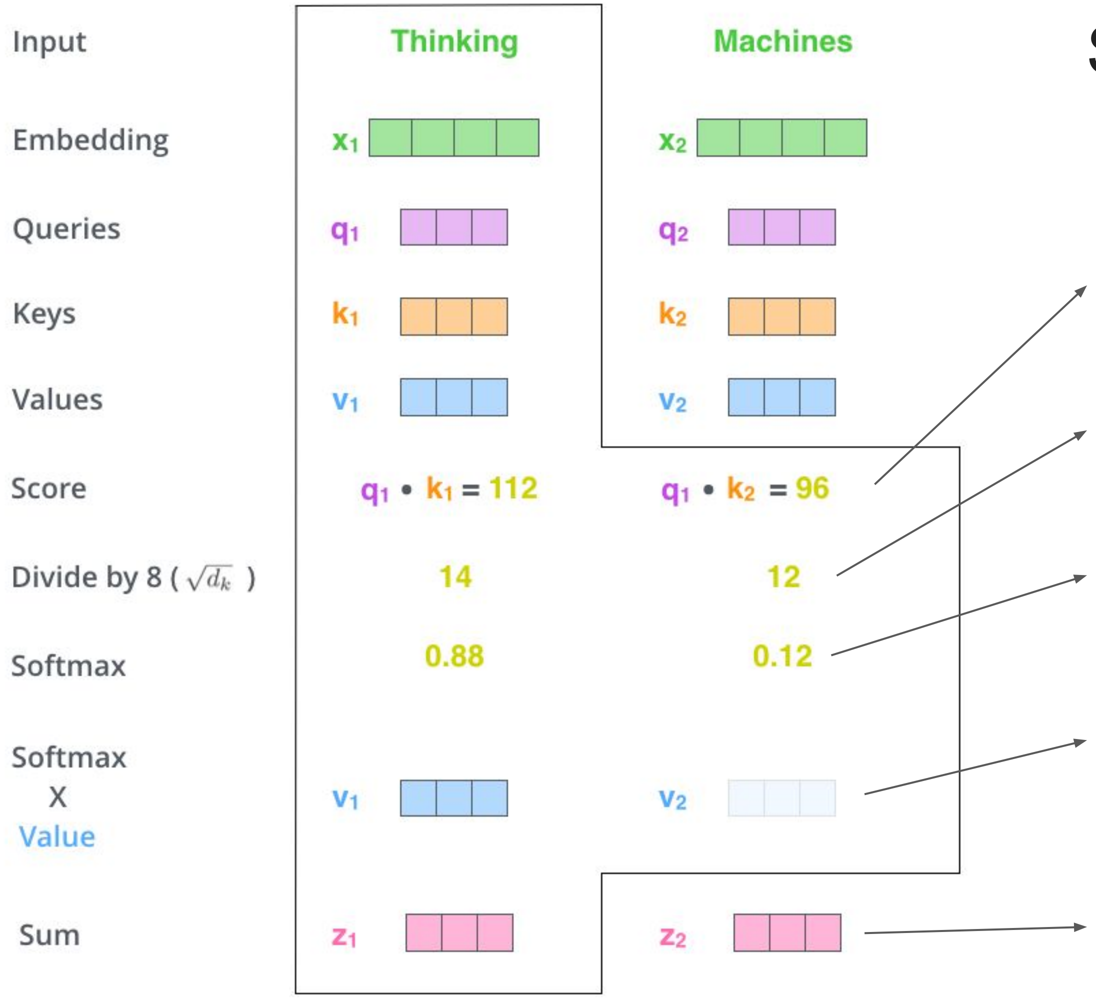
How is it implemented?

Query, Key, Value



Multiplying x1 by the W^Q weight matrix produces q1, the "query" vector associated with that word.
We end up creating a "query", a "key", and a "value" projection of each word in the input sentence.

Self-Attention in Detail



How much focus to place on other parts of the input sentence while encoding a word

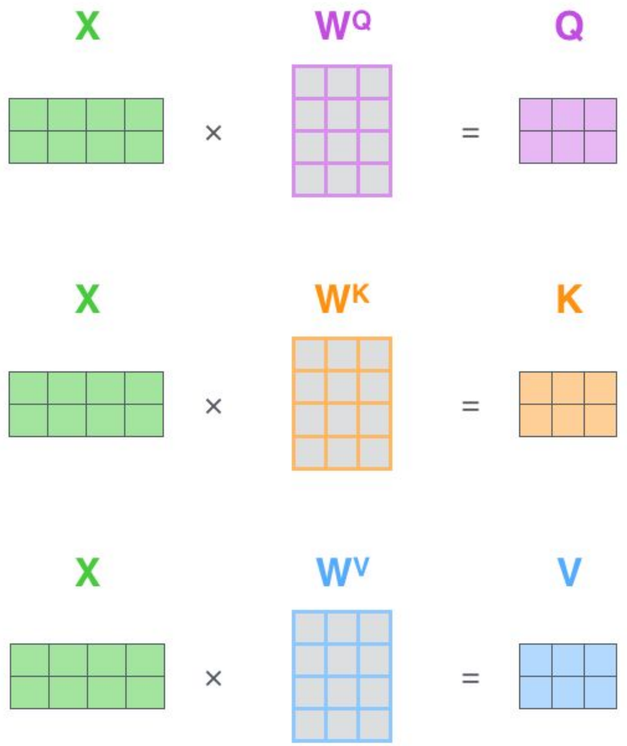
Regularization by the dimension of the key vector (more stable gradients)

How much each word will be expressed at this position

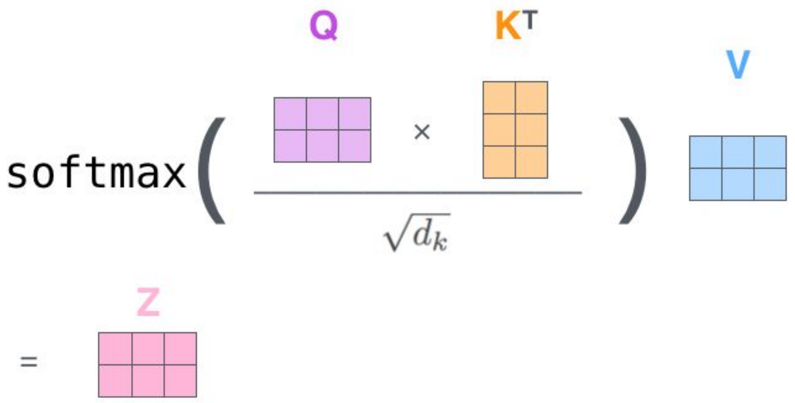
Keep intact the values of the word(s) we want to focus on, and drown-out irrelevant words

Sum up the weighted value vectors (output of the self-attention layer at this position)

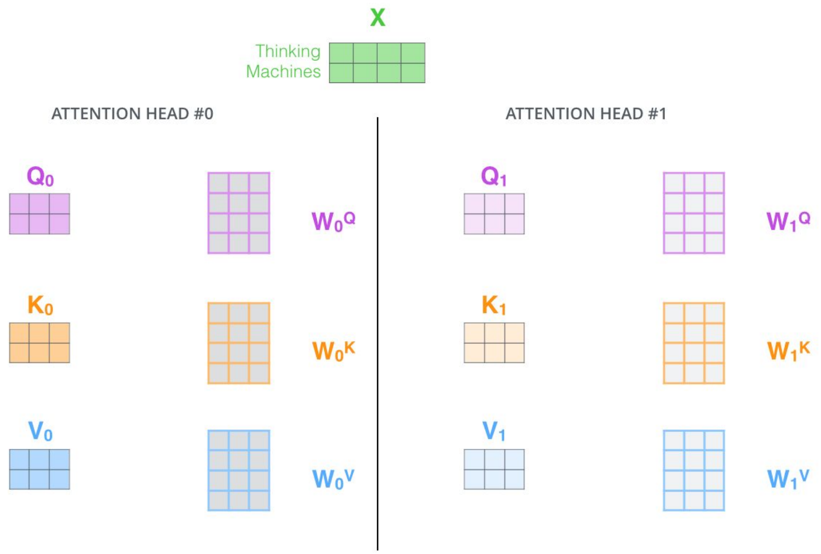
Self-Attention as Matrix Calculation



Self-attention is calculated in matrix form for faster processing

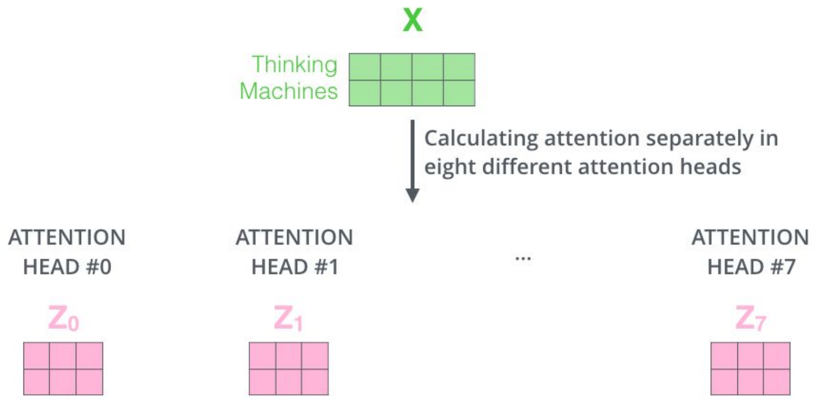


Multi-head Attention



Self-attention with 8 heads:

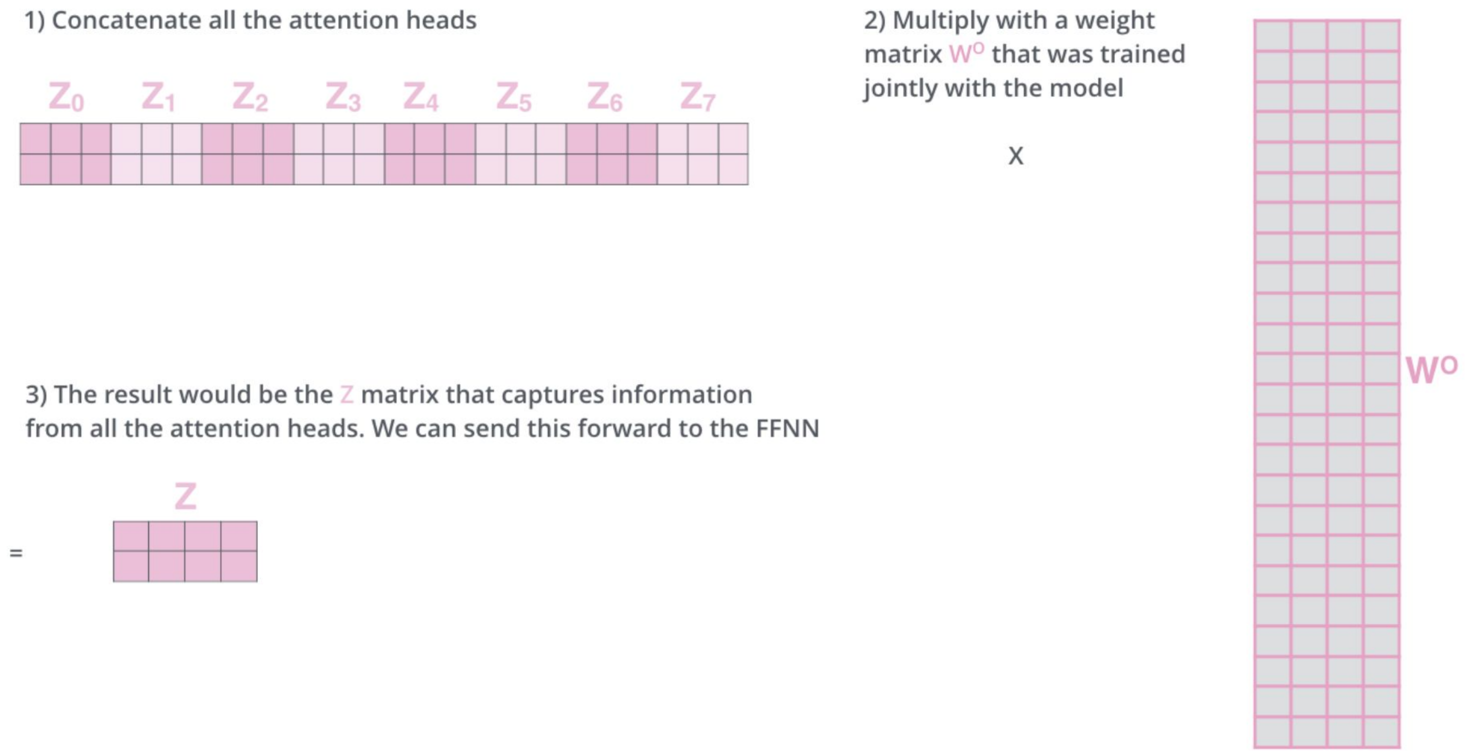
- eight different times with different weight matrices, we end up with eight different Z matrices



Multi-heads benefits:

- It expands the model's ability to focus on different positions.
- It gives the attention layer multiple "representation subspaces"

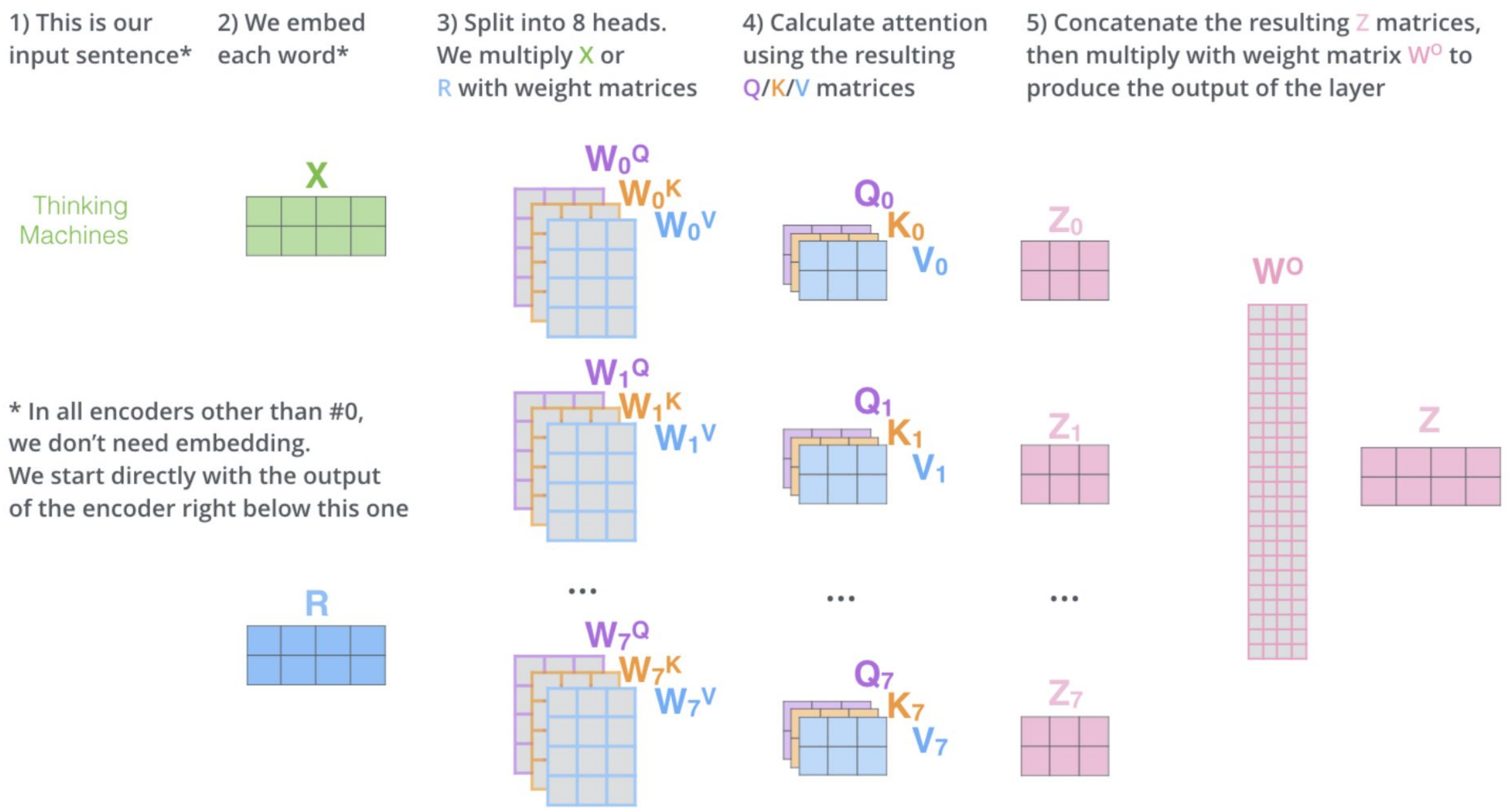
Concatenating Multi-head Attention


1) Concatenate all the attention heads

2) Multiply with a weight matrix W^O that was trained jointly with the model

3) The result would be the Z matrix that captures information from all the attention heads. We can send this forward to the FFNN

Putting it all together



1) This is our input sentence*

2) We embed each word*

3) Split into 8 heads. We multiply X or R with weight matrices

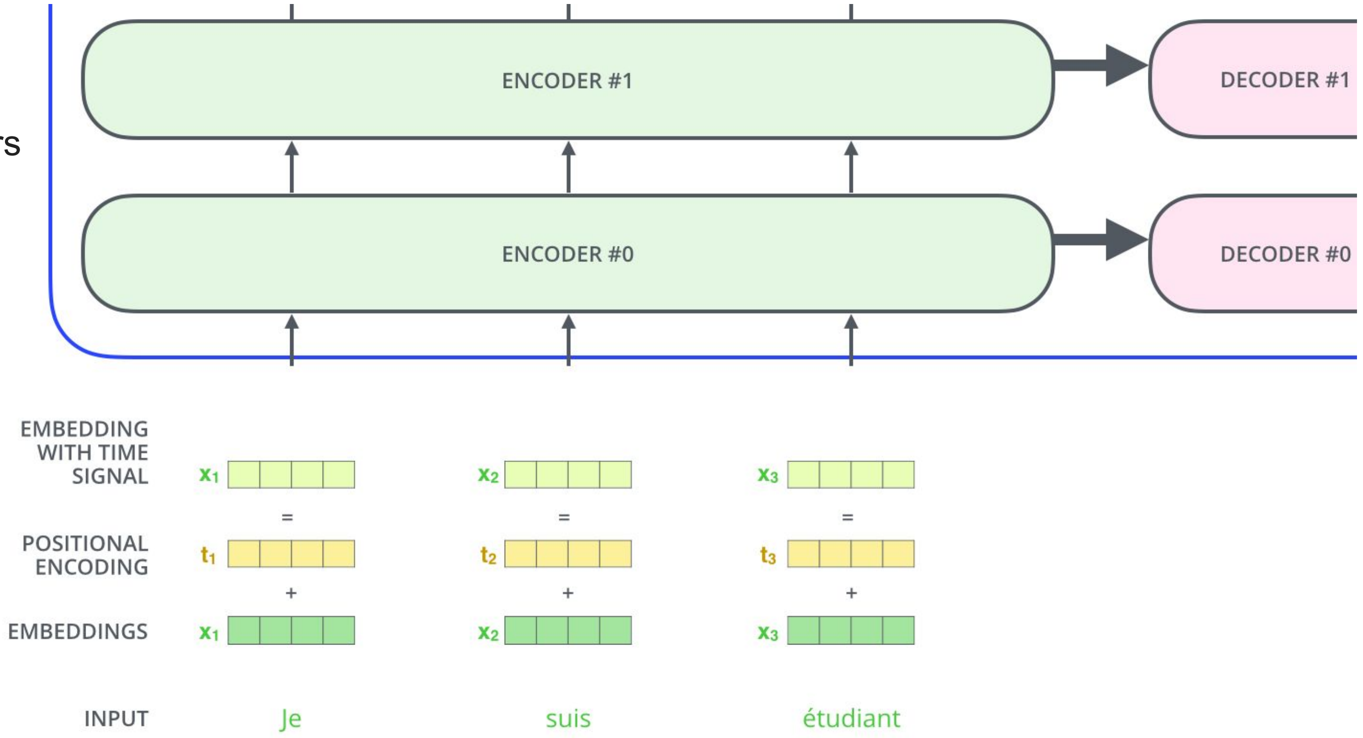
4) Calculate attention using the resulting Q/K/V matrices

5) Concatenate the resulting Z matrices, then multiply with weight matrix W^O to produce the output of the layer

* In all encoders other than #0, we don't need embedding. We start directly with the output of the encoder right below this one

Representing The Order of The Sequence Using Positional Encoding

Adding meaningful distances between the embedding vectors



Positional Encoding

- Positions can’t be encoded by numbers because sequences of different lengths can’t be trained this way
- Instead use a periodic function like sin, cos to create the embedding

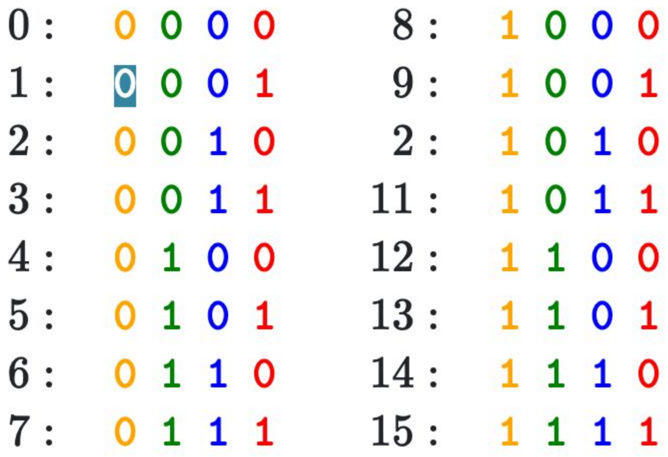
$$\vec{p_t}^{(i)} = f(t)^{(i)} := \begin{cases} \sin(\omega_k . t), & \text{if } i = 2k \\ \cos(\omega_k . t), & \text{if } i = 2k+1 \end{cases}$$

$$\omega_k = \frac{1}{10000^{2k/d}}$$

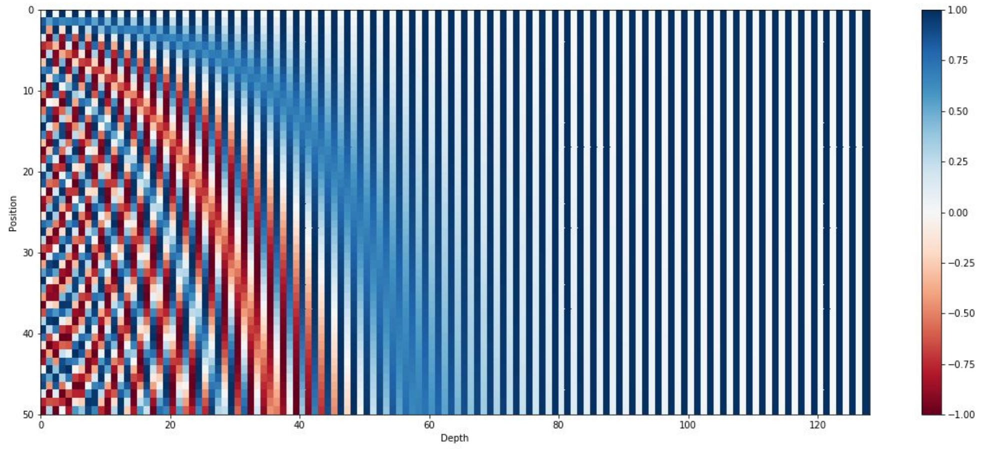
Binary vs Positional Encoding

Why not using Binary Encoding

- Positional encoding scales better with longer sequences due to its fixed dimensionality regardless of the sequence length.
- Binary encoding requires more bits as the sequence length increases, which might not be efficient for very long sequences.



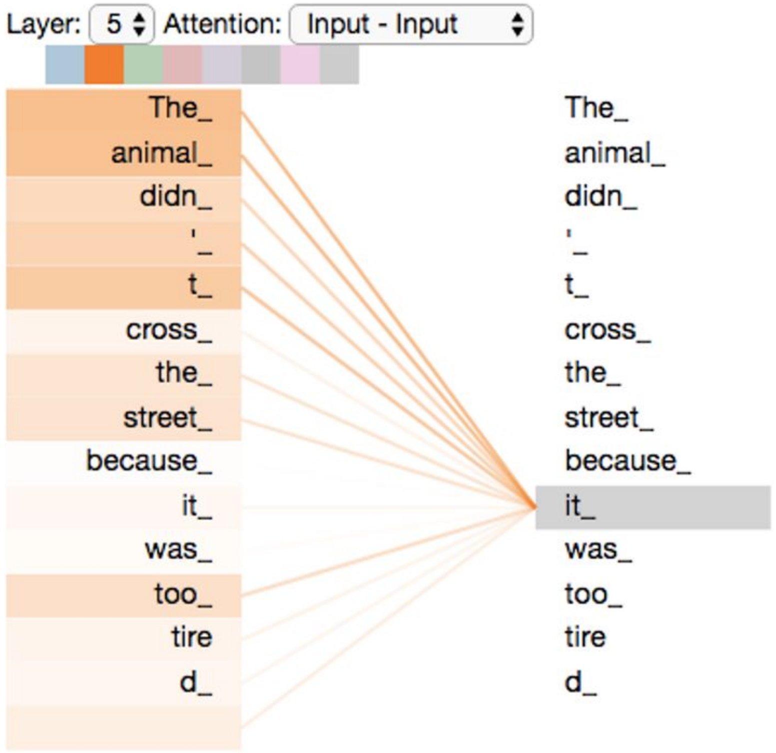
0 :	0	0	0	0	8 :	1	0	0	0
1 :	0	0	0	1	9 :	1	0	0	1
2 :	0	0	1	0	2 :	1	0	1	0
3 :	0	0	1	1	11 :	1	0	1	1
4 :	0	1	0	0	12 :	1	1	0	0
5 :	0	1	0	1	13 :	1	1	0	1
6 :	0	1	1	0	14 :	1	1	1	0
7 :	0	1	1	1	15 :	1	1	1	1



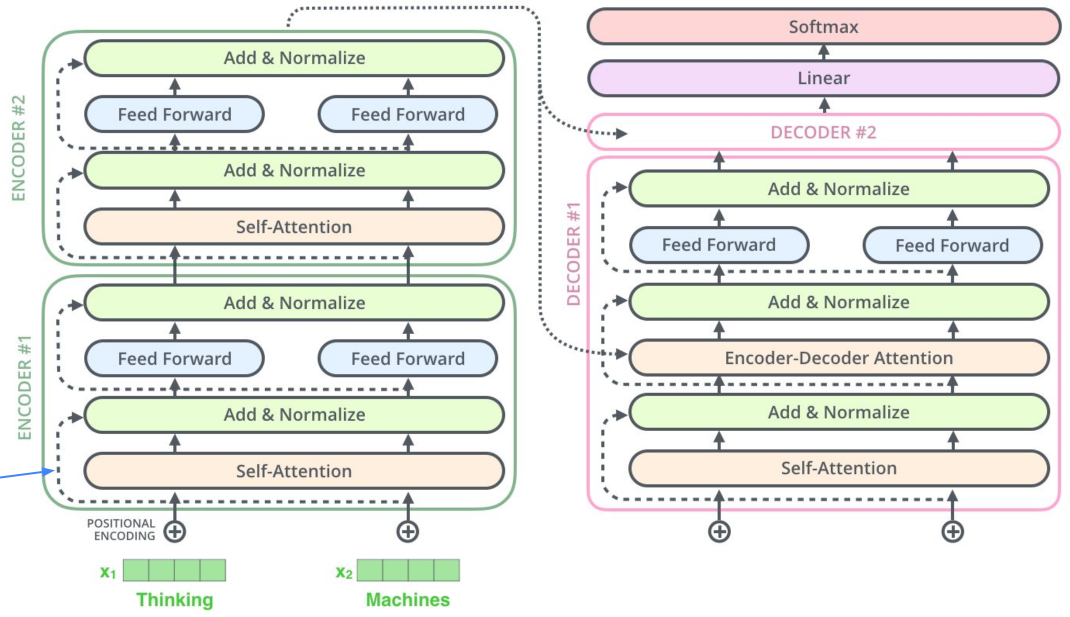
Attention maps over tokens

"The animal didn't cross the street because it was too tired"

Self-attention helps a word gather information from other positions in the input sequence to enhance its representation.



Transformer architecture

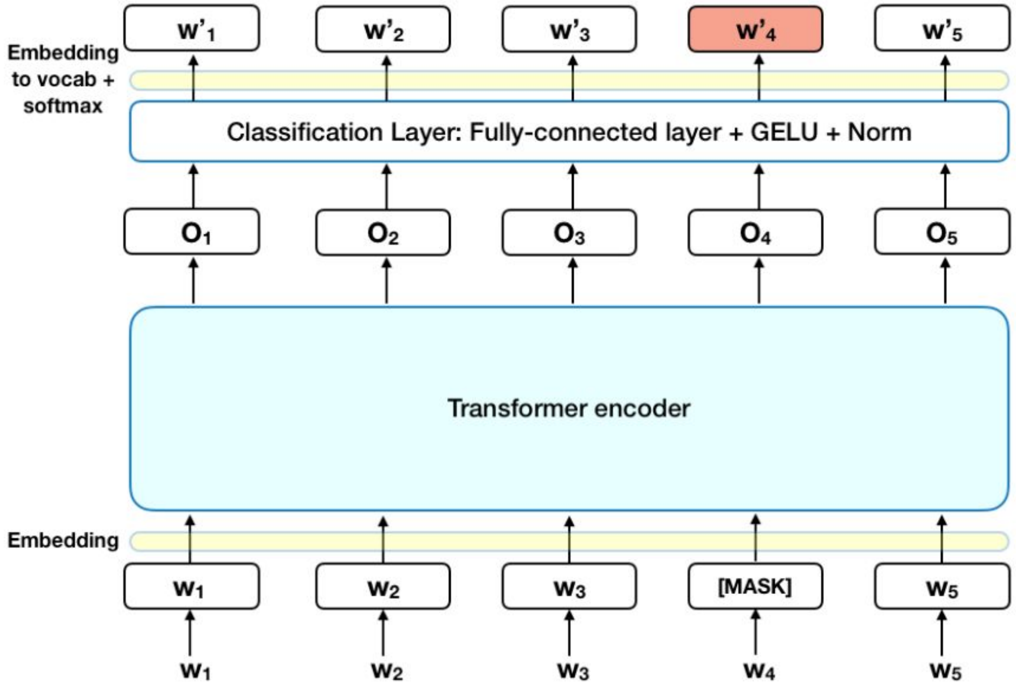


Residual connection

Learning language with Transformer-based models

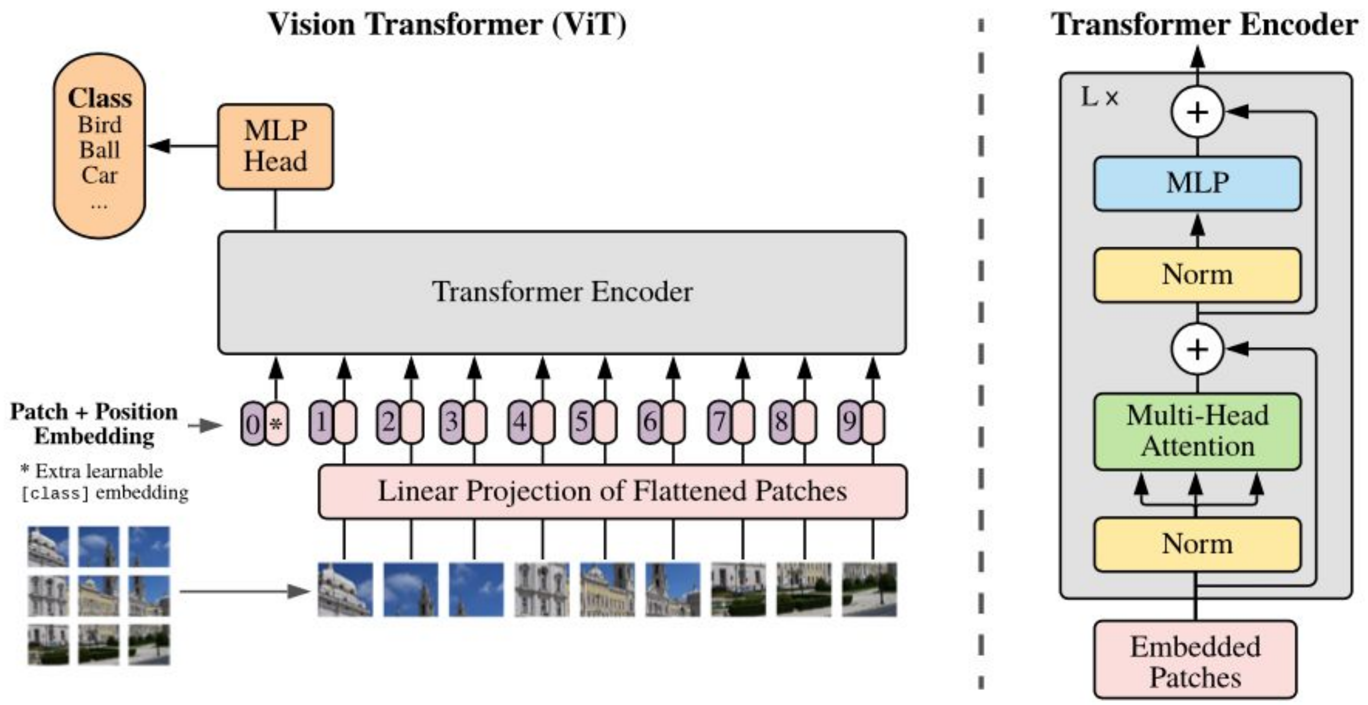
BERT: Bidirectional Encoder Representations from Transformers

- The meaning is heavily dependent on the context
- Learning a language via predicting masked tokens (words/characters)
- Contextualized prediction


w'1
w'2
w'3
w'4
w'5
Embedding to vocab + softmax
Classification Layer: Fully-connected layer + GELU + Norm
O1
O2
O3
O4
O5
Transformer encoder
Embedding
w1
w2
w3
[MASK]
w5
w1
w2
w3
w4
w5

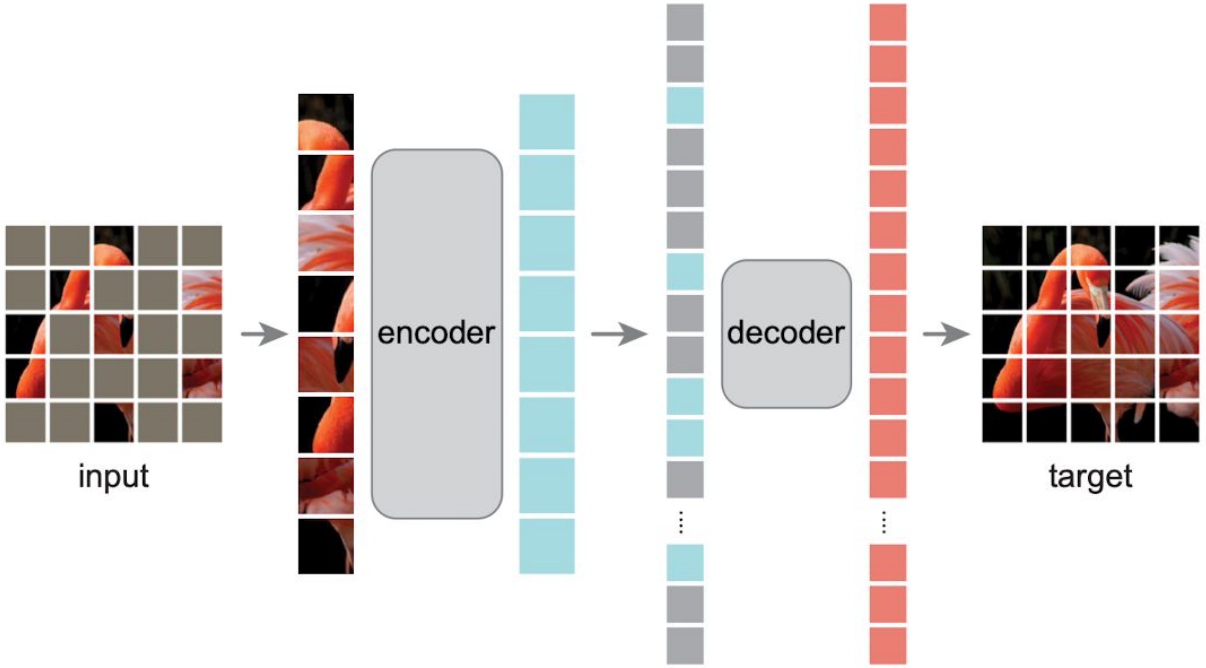
Can we apply this to other domains?

Vision Transformer (VIT)



How can we turn this into an unsupervised task?

Masked Autoencoder (MAE)



Foundation Models

Large Models + Big data = Foundation Models

Key Characteristics:

- **Scale**: Large models that are trained on extensive datasets.
- **Unsupervised**: Trained to predict the data itself
- **Transfer Learning**: Use pre-trained knowledge for task-specific applications.

Examples:

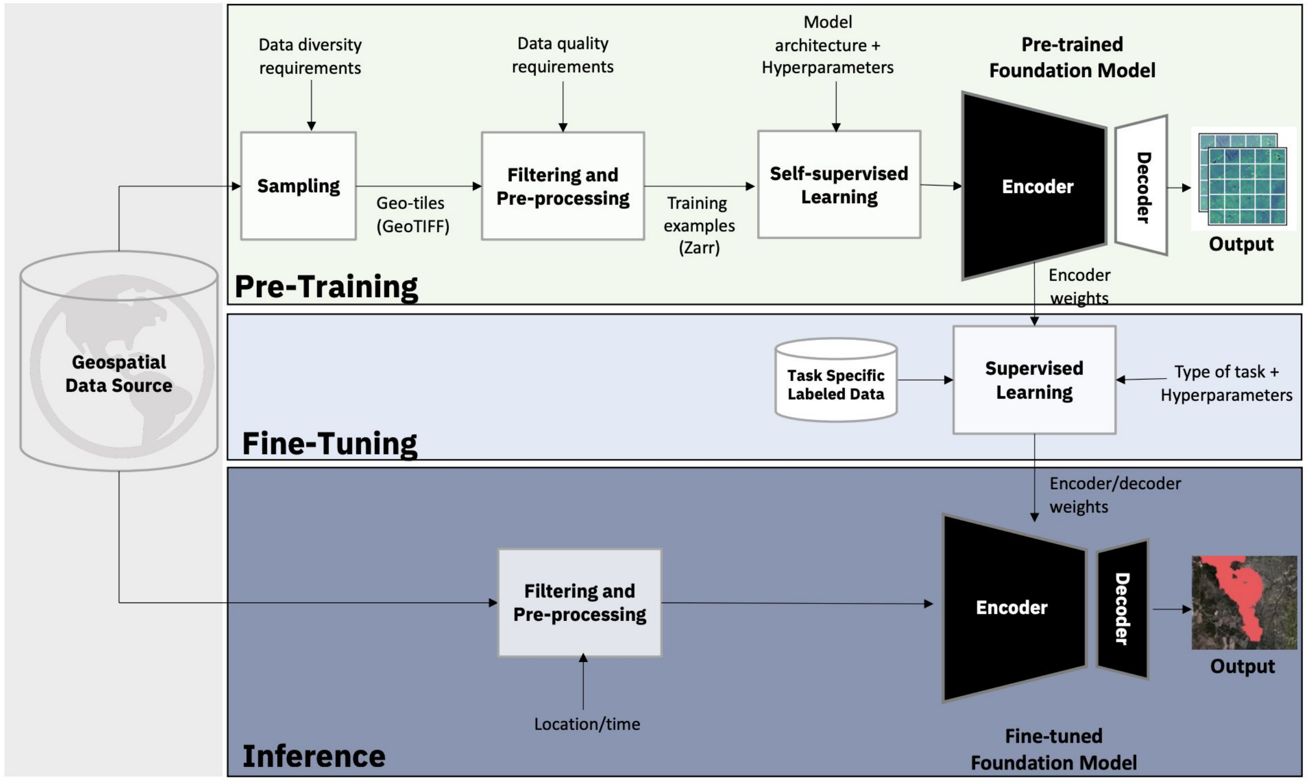
1. **GPT-3 (OpenAI)**:
   - **Parameters**: 175 billion
   - **Dataset Size**: 410 billion tokens (source: Common Crawl, WebText2, Books1, Books2, Wikipedia)
2. **BERT (Google)**:
   - **Parameters**: 340 million (BERT Large)
   - **Dataset Size**: 3.3 billion tokens (source: BooksCorpus and English Wikipedia)
3. **CLIP (OpenAI)**:
   - **Parameters**: 428 million (Transformer + ViT-Large)
   - **Dataset Size**: 400 million image-text (source: Common Crawl)

Prithvi

A transformer-based geospatial foundational model

- Model size: 100 million parameters
- Dataset: 1TB of multispectral satellite imagery from the Harmonized Landsat-Sentinel 2 (HLS) dataset
- HLS dataset:
  - Spatial resolution: 30 m
  - Operational Land Imager (OLI) and Multi-Spectral Instrument (MSI) aboard the Landsat 8/9 and Sentinel-2 remote sensing satellites
  - Global observations of the land every 2–3 days.

Prithvi's framework


Data diversity requirements
Data quality requirements
Model architecture + Hyperparameters
Pre-trained Foundation Model
Sampling
Geo-tiles (GeoTIFF)
Filtering and Pre-processing
Training examples (Zarr)
Self-supervised Learning
Encoder
Decoder
Output
Pre-Training
Encoder weights
Geospatial Data Source
Task Specific Labeled Data
Supervised Learning
Type of task + Hyperparameters
Fine-Tuning
Encoder/decoder weights
Filtering and Pre-processing
Encoder
Decoder
Output
Location/time
Inference
Fine-tuned Foundation Model

Prithvi's framework

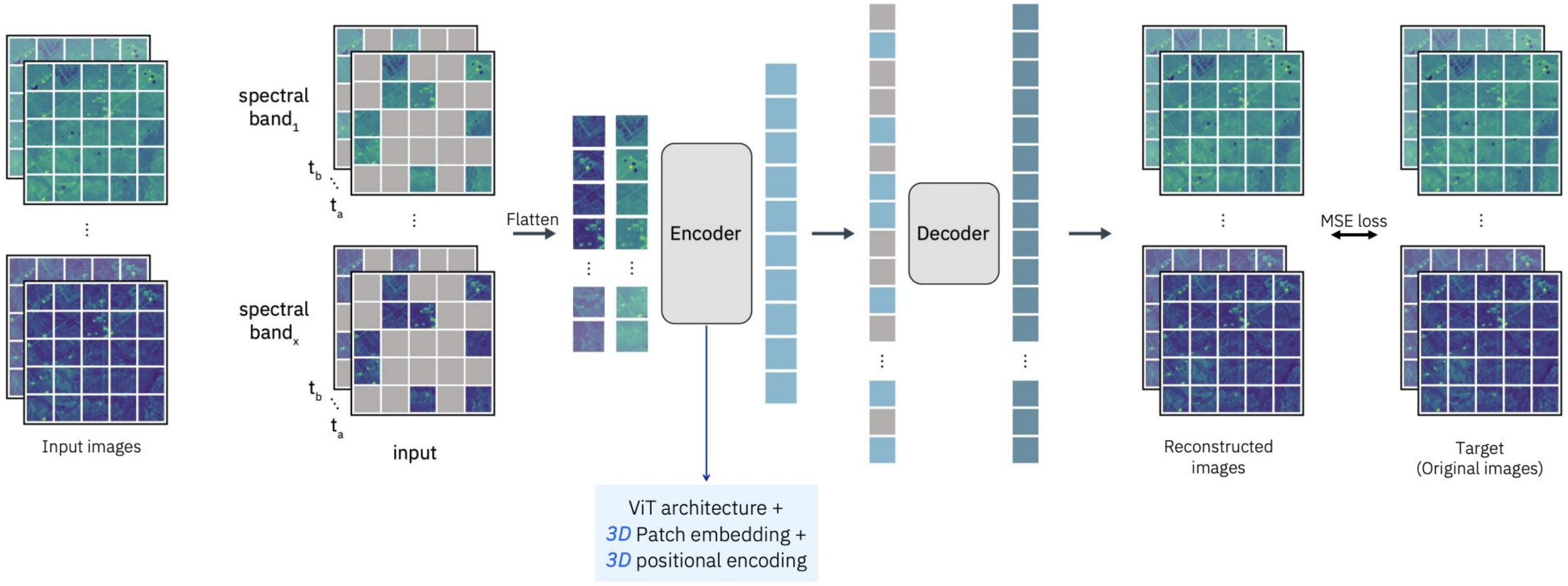


Fig. 3: The masked autoencoder (MAE) structure for pre-training Prithvi on large-scale multi-temporal and multi-spectral satellite images.

Fine-tuning

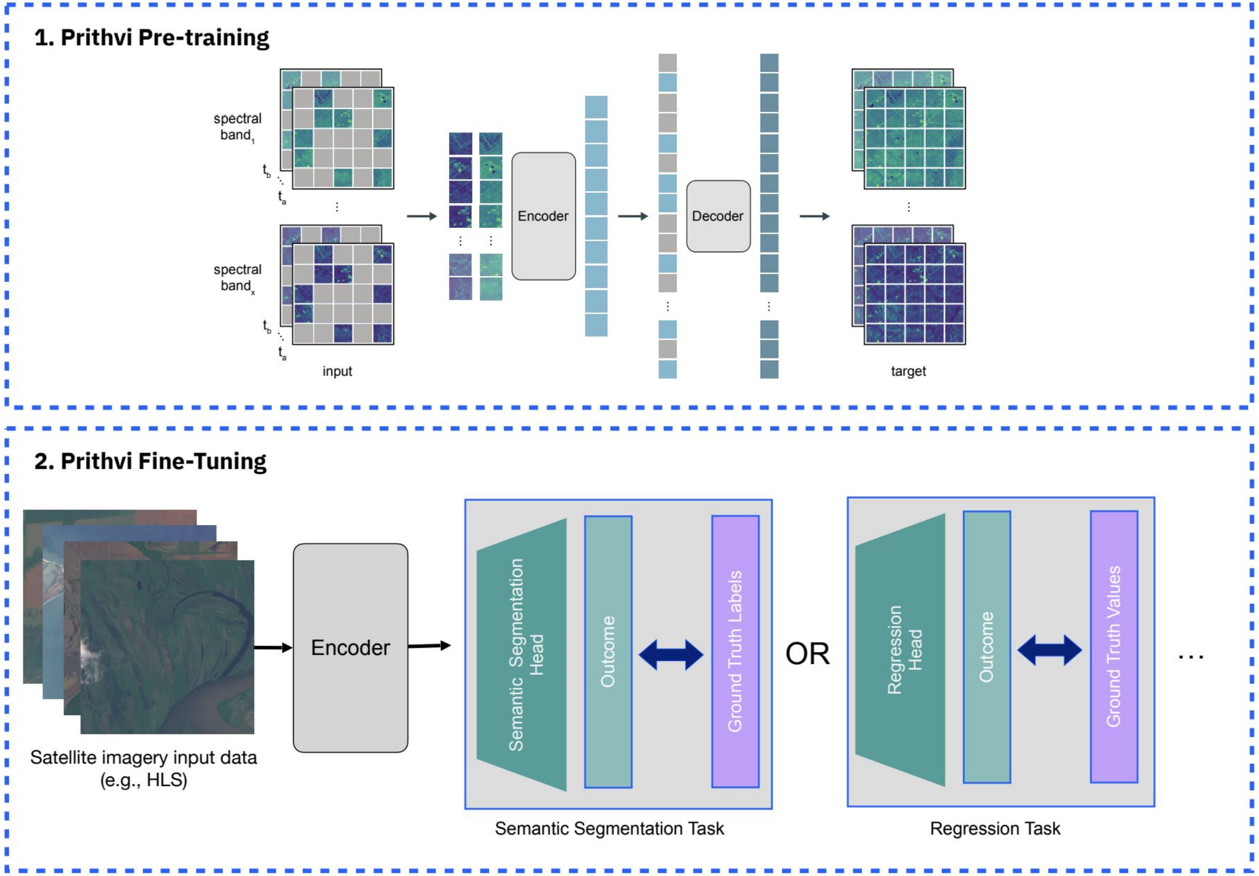
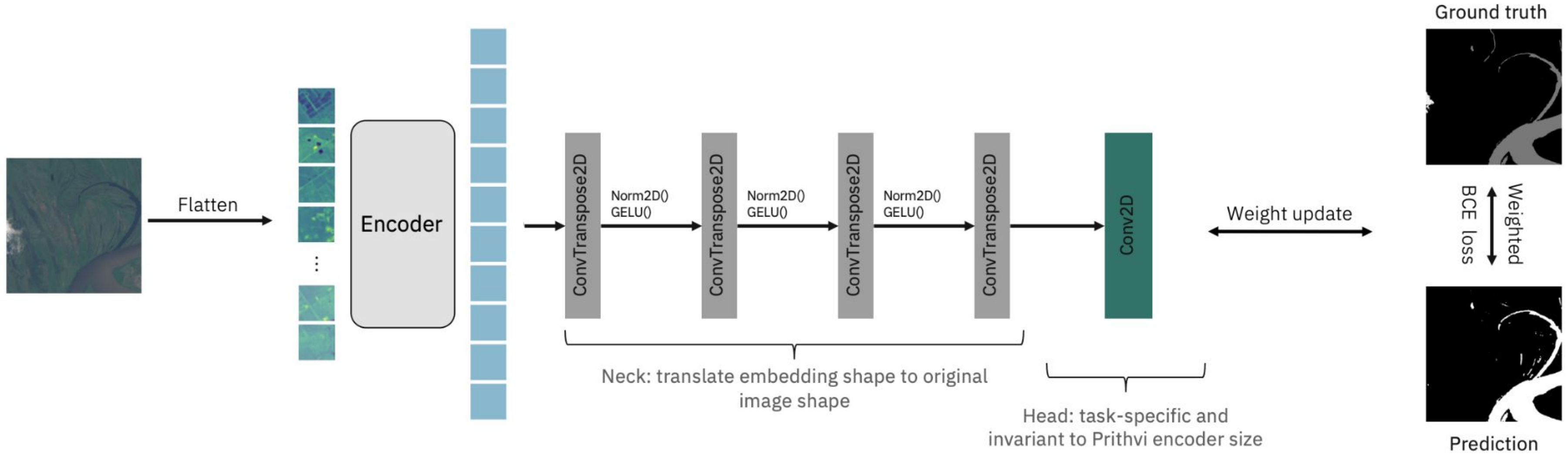
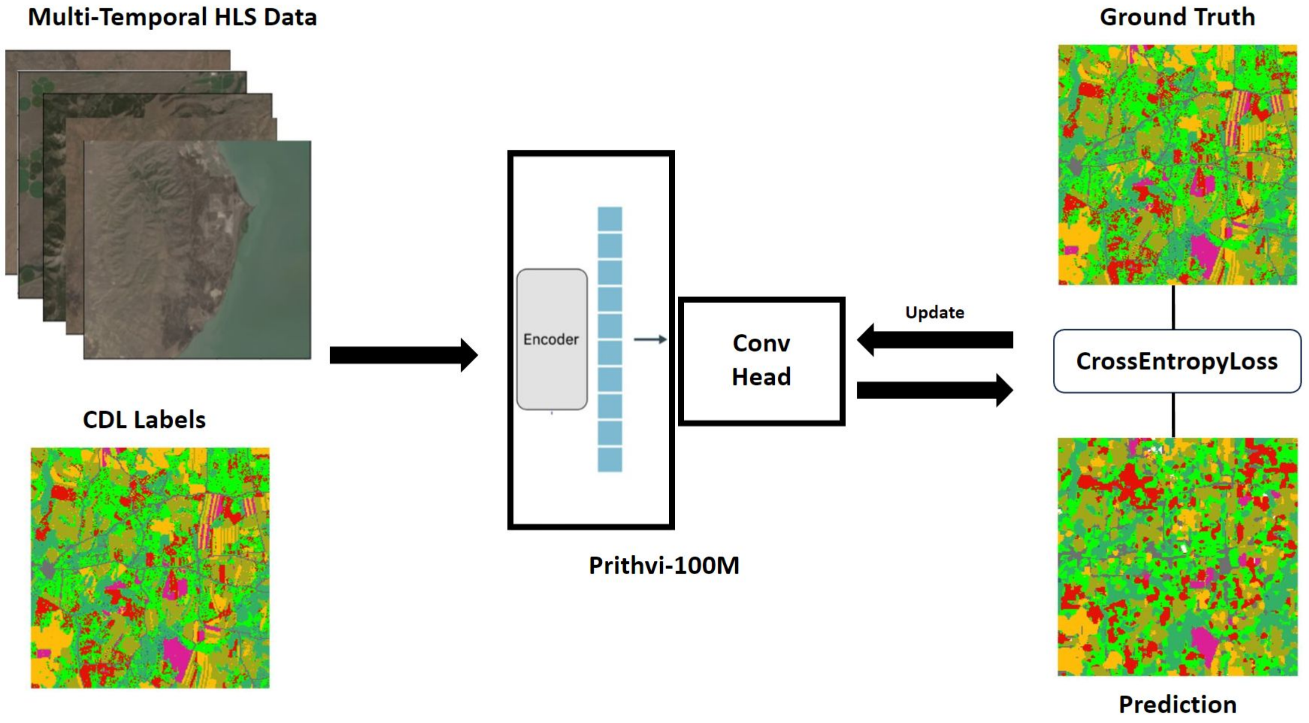


Fig. 4: Pre-training and fine-tuning in Prithvi for various types of downstream tasks.

Flood detection



Crop classification



Conclusion

- Foundation Models: large models that are trained on extensive datasets.
- Advantage: A model that already understands the “basics” about the data
- Few-shot learning: Much smaller number of samples to tune it for your application
- Fine-tuning: Use the model’s knowledge for a downstream application

Tutorial

Let’s explore Prithvi and its fine-tuned models:

1) Predict masked regions of an image (pretrained)
2) Predict regions with likelihood of flooding (fine-tuned)
3) Predict different types of crop (fine-tuned)
4) BONUS: download an image for your hometown and do the prediction tasks above [did the model get it right?]