# Deep Learning for Geospatial Data

Maria Szlasa

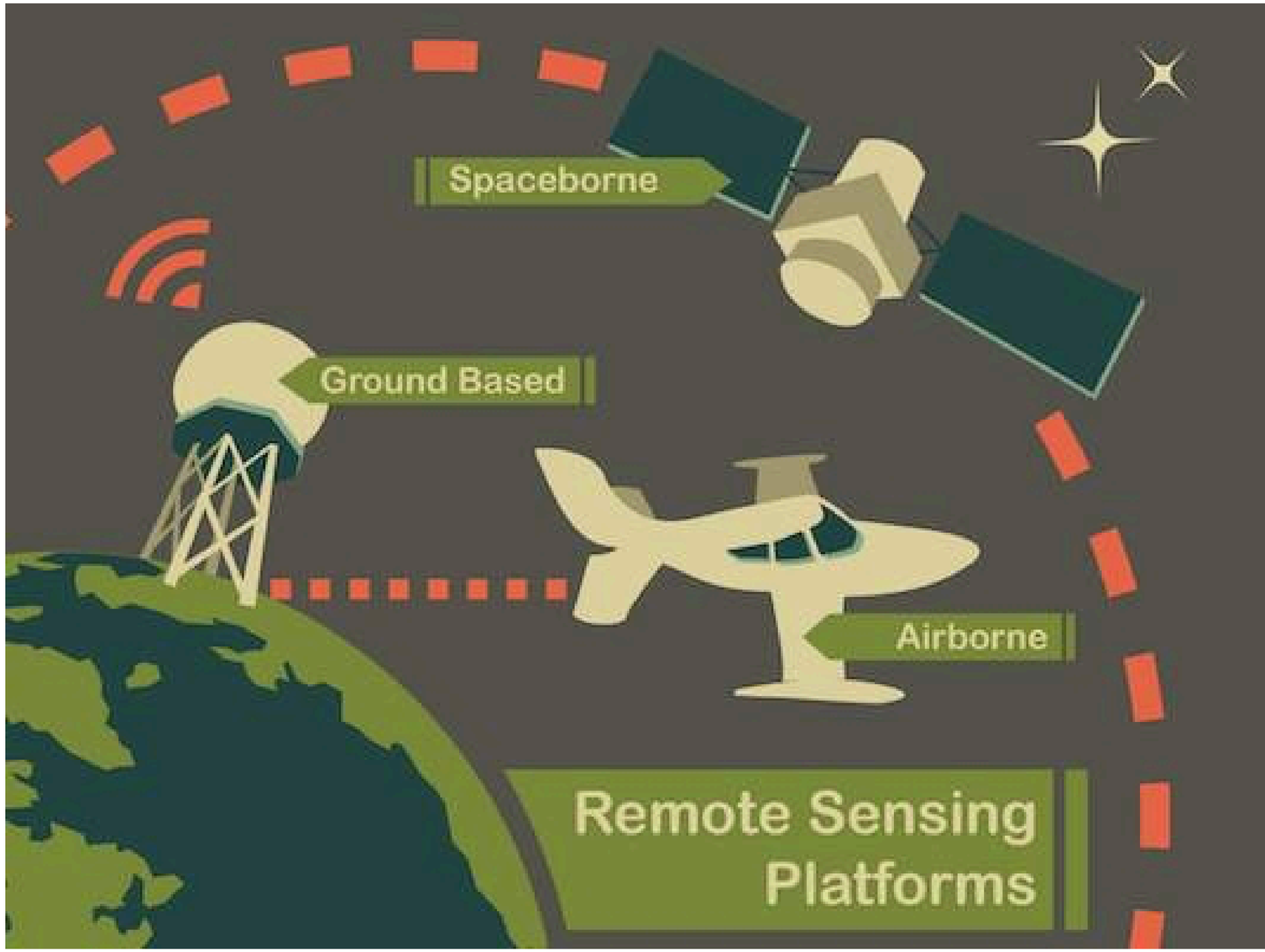
# Agenda

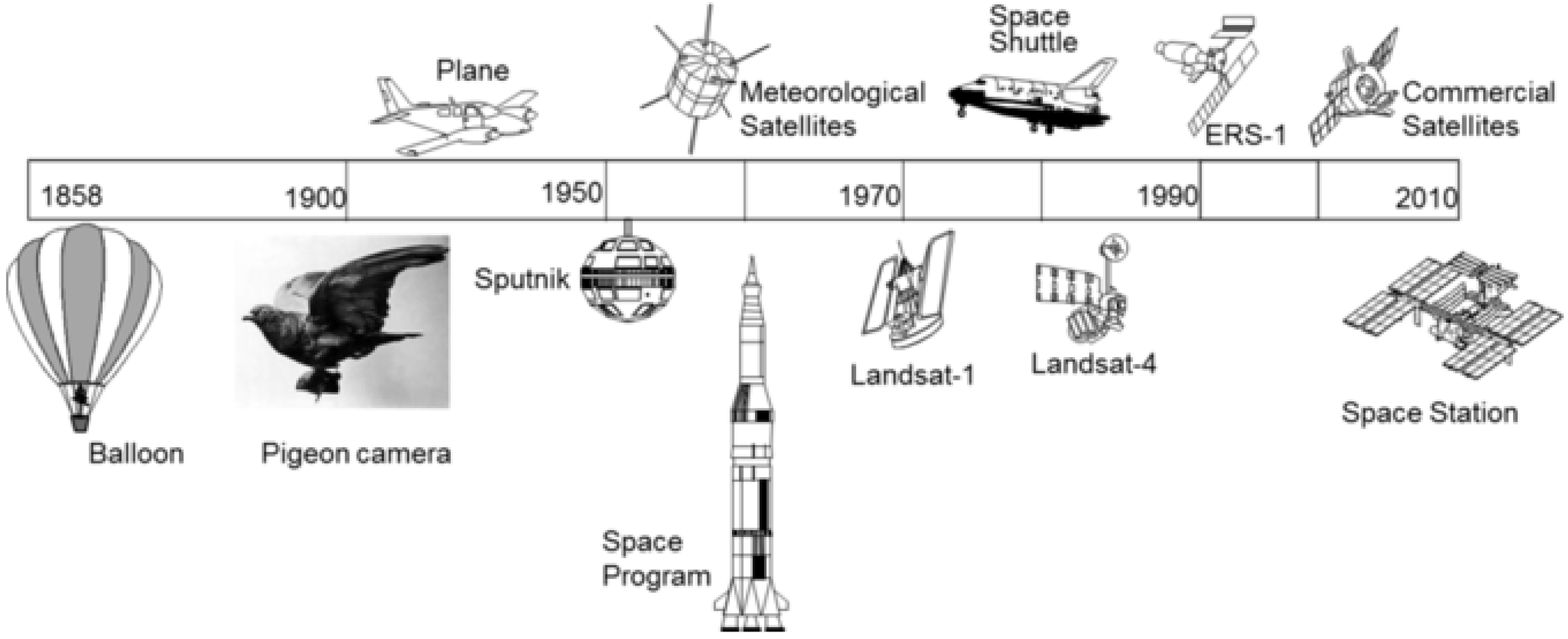
1. Introduction to Remote Sensing
2. Data
3. Applications of Deep Learning in Geospatial Data
4. Basic Models
    a. UNet
    b. Vision Transformer
5. Satellite Image Time Series
    a. SatMAE
    b. Presto
    c. GraphCast
6. Case Studies

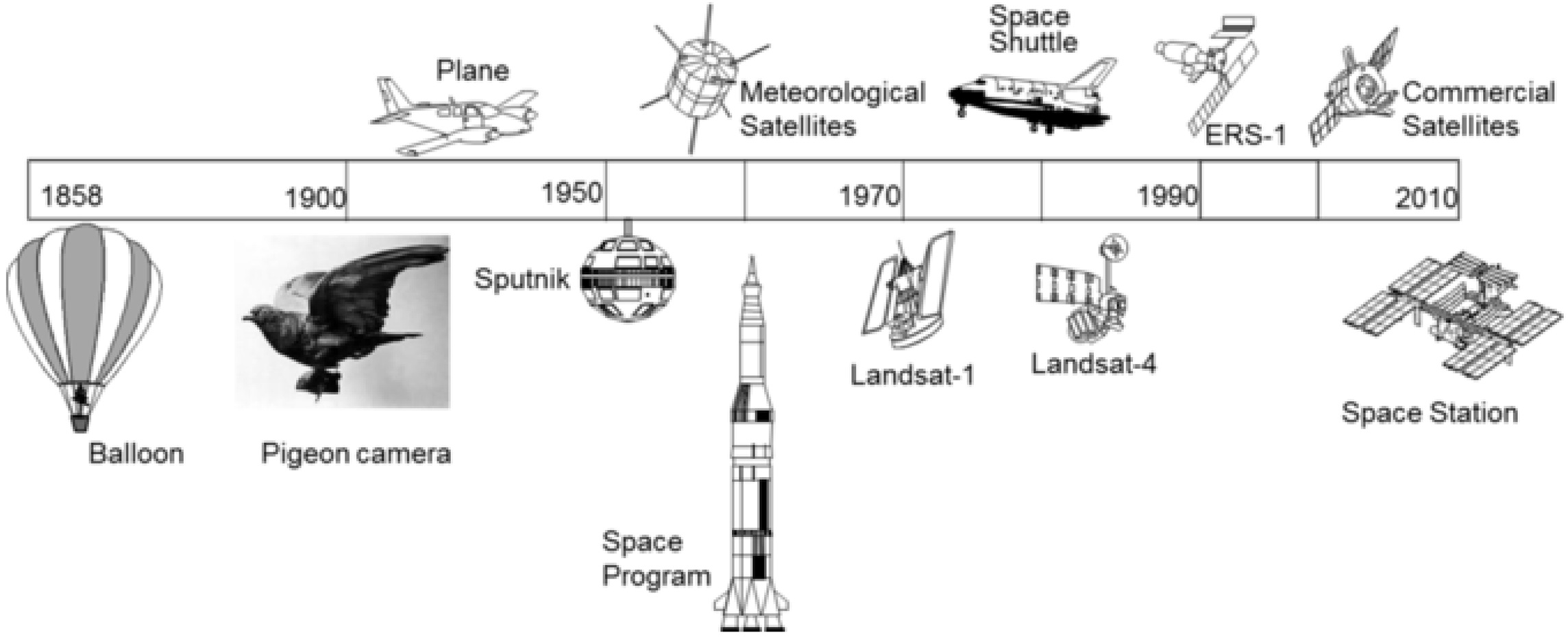
# Introduction to Remote Sensing

# What is remote sensing?

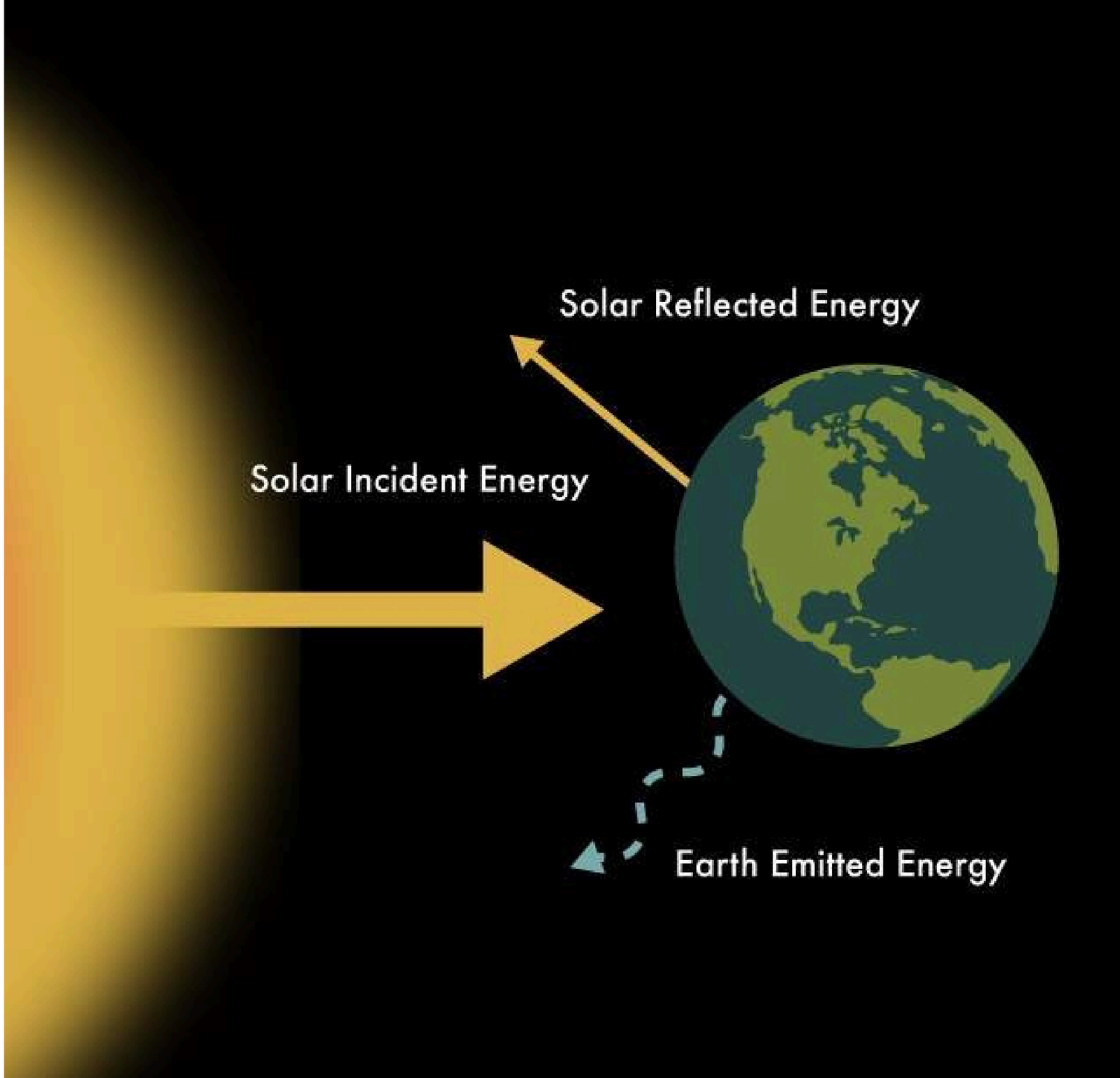
Remote sensing is obtaining information about an object from a distance.

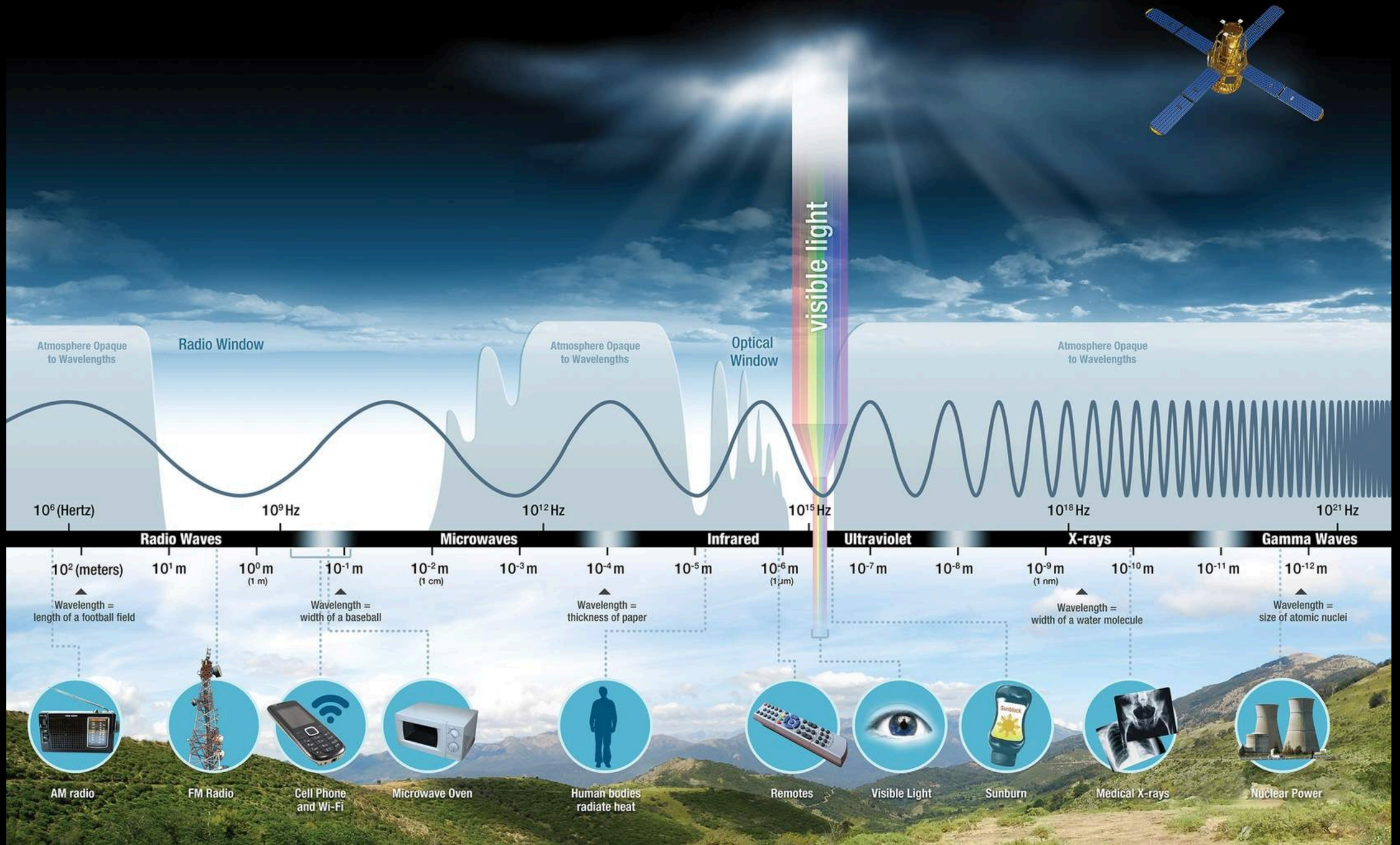



Plane
Meteorological Satellites
Space Shuttle
ERS-1
Commercial Satellites
1858
1900
1950
1970
1990
2010
Balloon
Pigeon camera
Sputnik
Space Program
Landsat-1
Landsat-4
Space Station

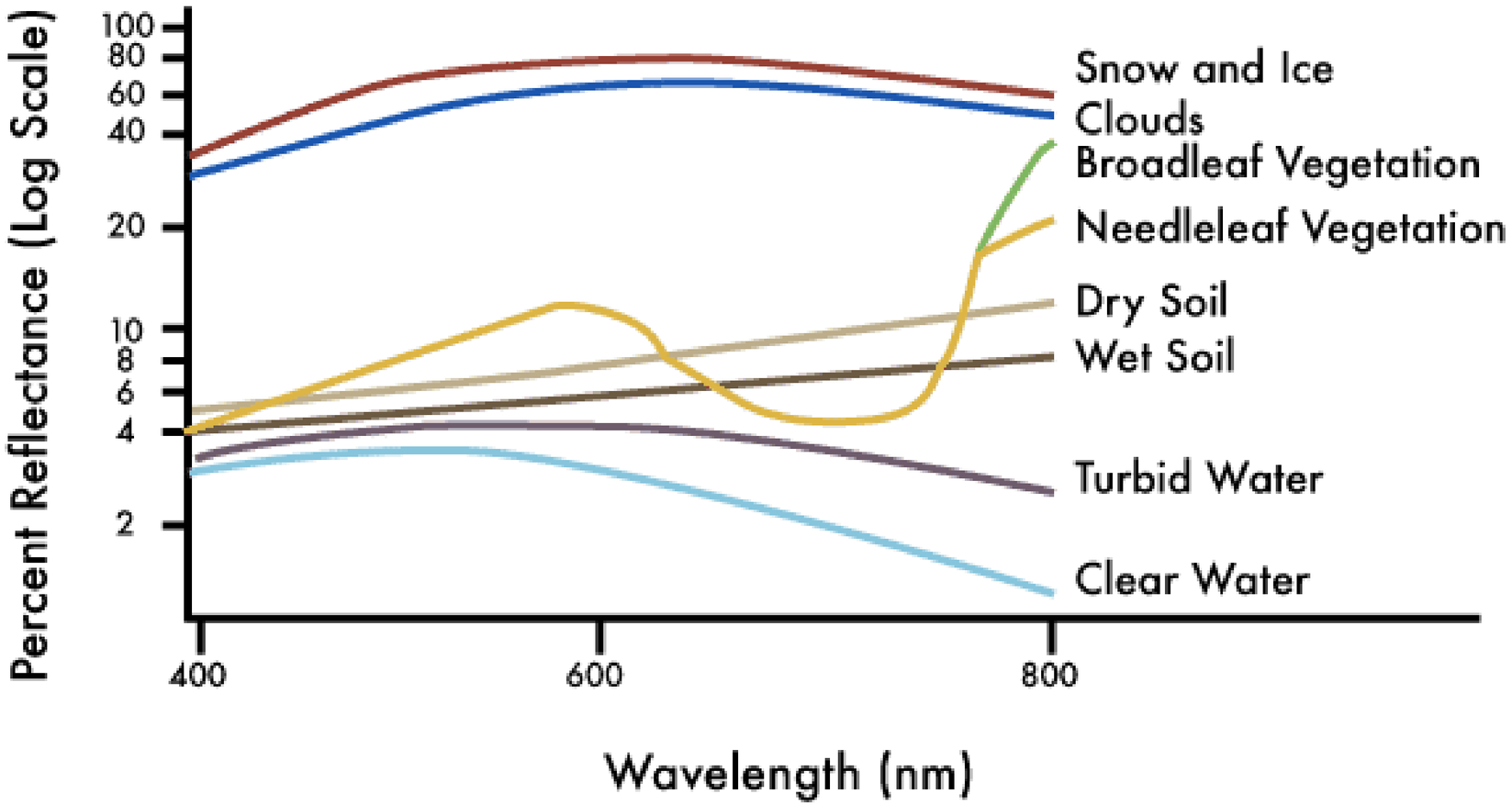

Plane
Meteorological Satellites
Space Shuttle
ERS-1
Commercial Satellites
1858
1900
1950
1970
1990
2010
Balloon
Pigeon camera
Sputnik
Space Program
Landsat-1
Landsat-4
Space Station

# Measuring Properties of the Earth-Atmosphere System from Space




Atmosphere Opaque to Wavelengths
Radio Window
Atmosphere Opaque to Wavelengths
Optical Window
visible light
Atmosphere Opaque to Wavelengths
10^6 (Hertz)
10^9 Hz
10^12 Hz
10^15 Hz
10^18 Hz
10^21 Hz
Radio Waves
Microwaves
Infrared
Ultraviolet
X-rays
Gamma Waves
10^2 (meters)
10^1 m
10^0 m (1 m)
10^-1 m
10^-2 m (1 cm)
10^-3 m
10^-4 m
10^-5 m
10^-6 m (1 µm)
10^-7 m
10^-8 m
10^-9 m (1 nm)
10^-10 m
10^-11 m
10^-12 m
Wavelength = length of a football field
Wavelength = width of a baseball
Wavelength = thickness of paper
Wavelength = width of a water molecule
Wavelength = size of atomic nuclei
AM radio
FM Radio
Cell Phone and Wi-Fi
Microwave Oven
Human bodies radiate heat
Remotes
Visible Light
Sunblock
Sunburn
Medical X-rays
Nuclear Power

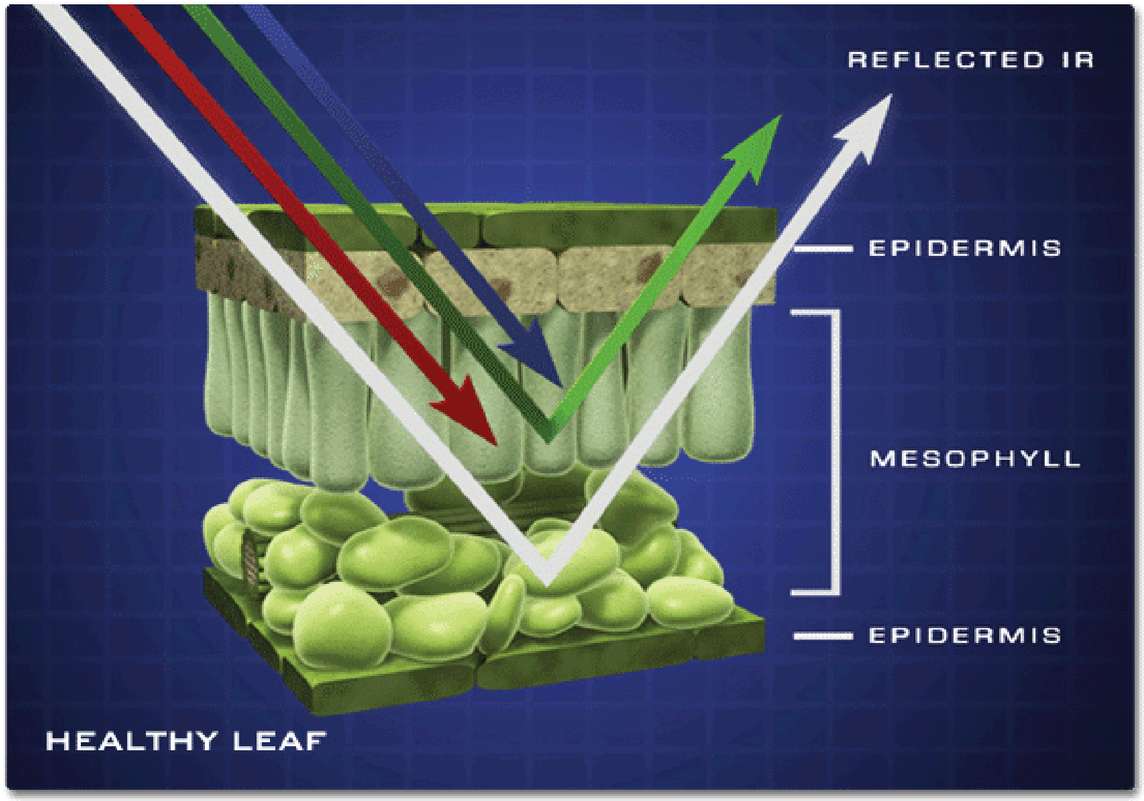
# Spectral Signatures



# Spectral Signatures

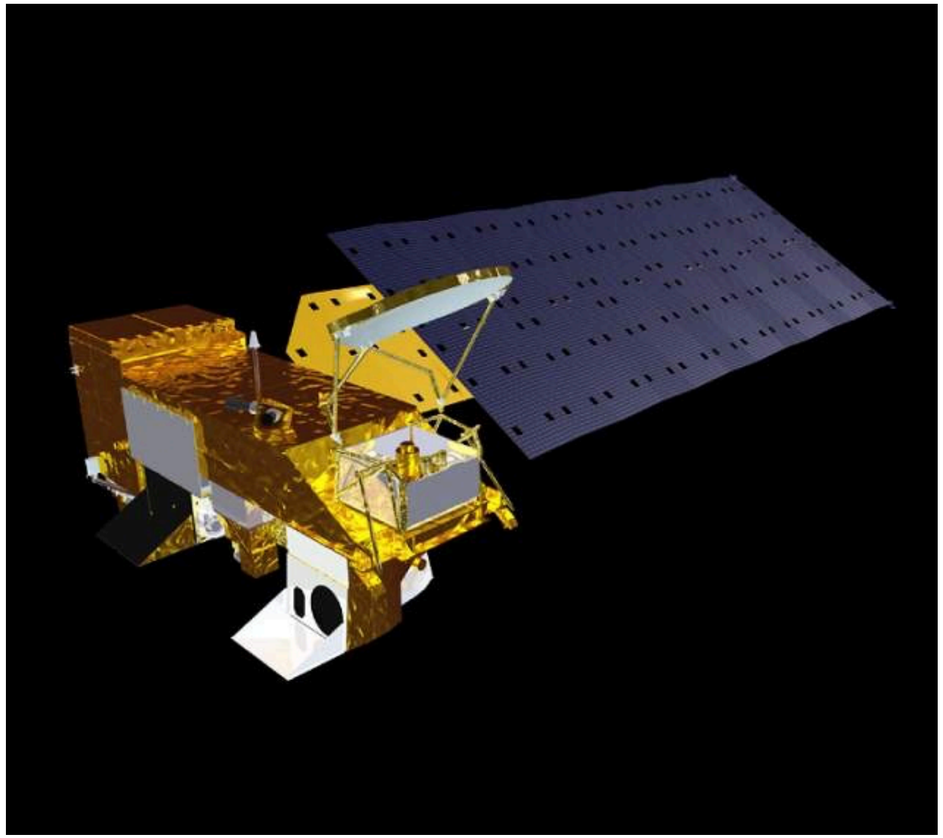
## Vegetation

Healthy vegetation absorbs blue and red wavelengths, but reflects green and infrared.

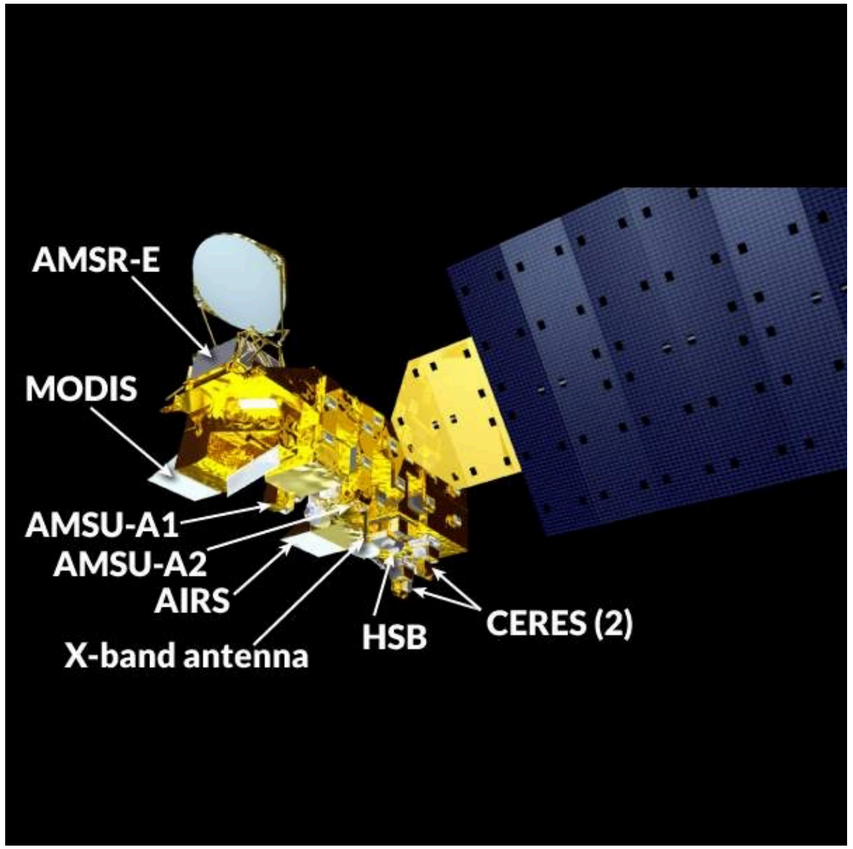


# Satellites and Sensors

# Satellites vs. Sensors

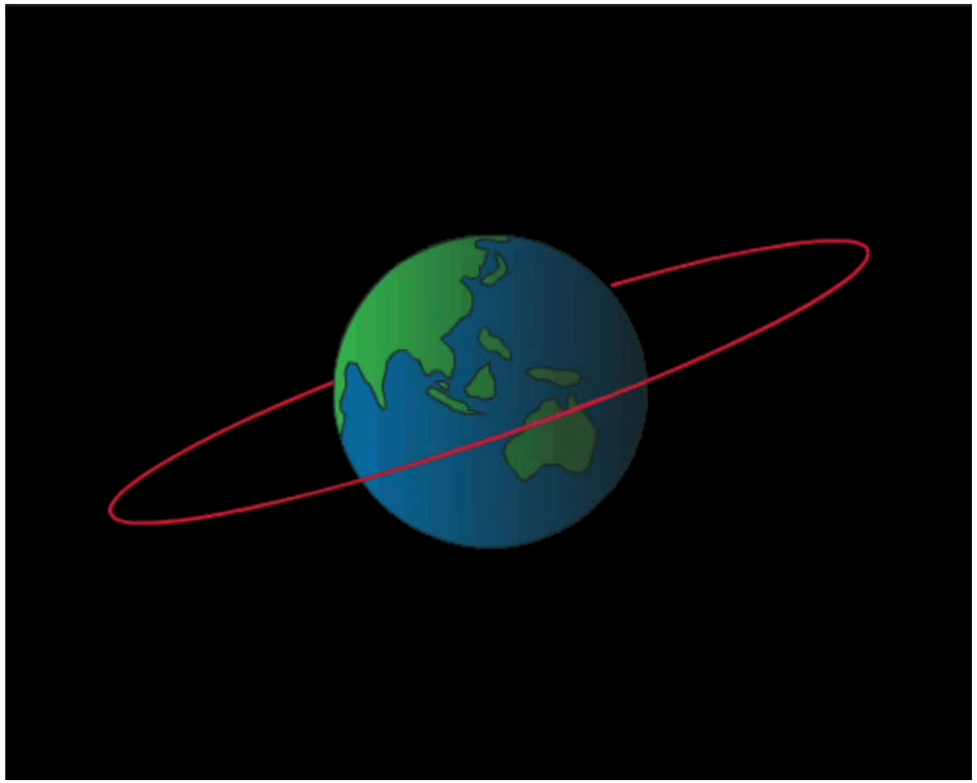


Satellite

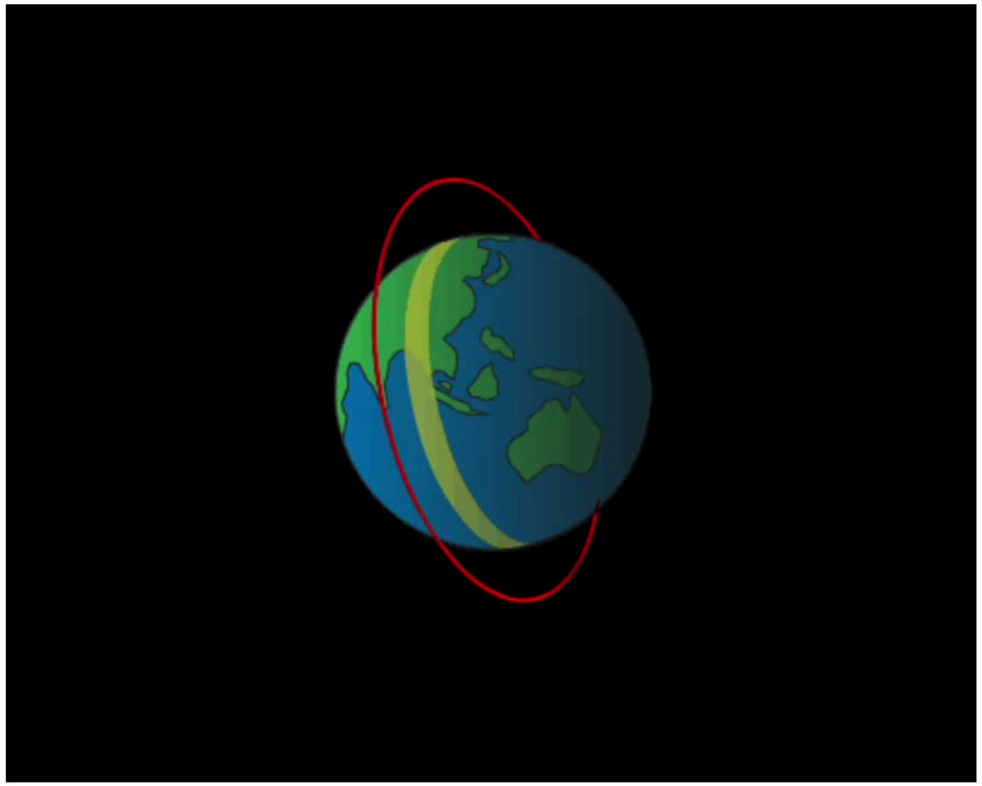



Sensors

# Types of Satellite Orbits



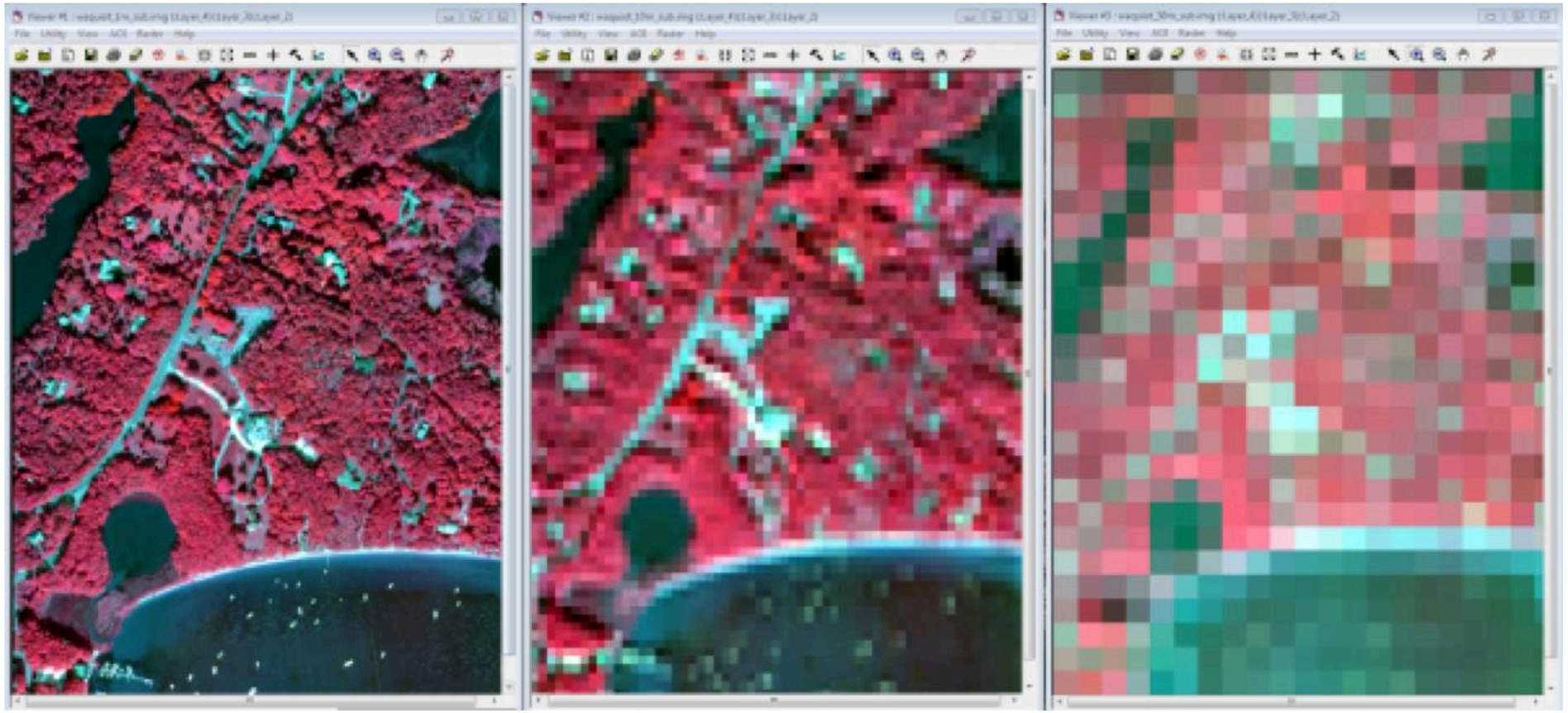
Geostationary



Low Earth Orbit

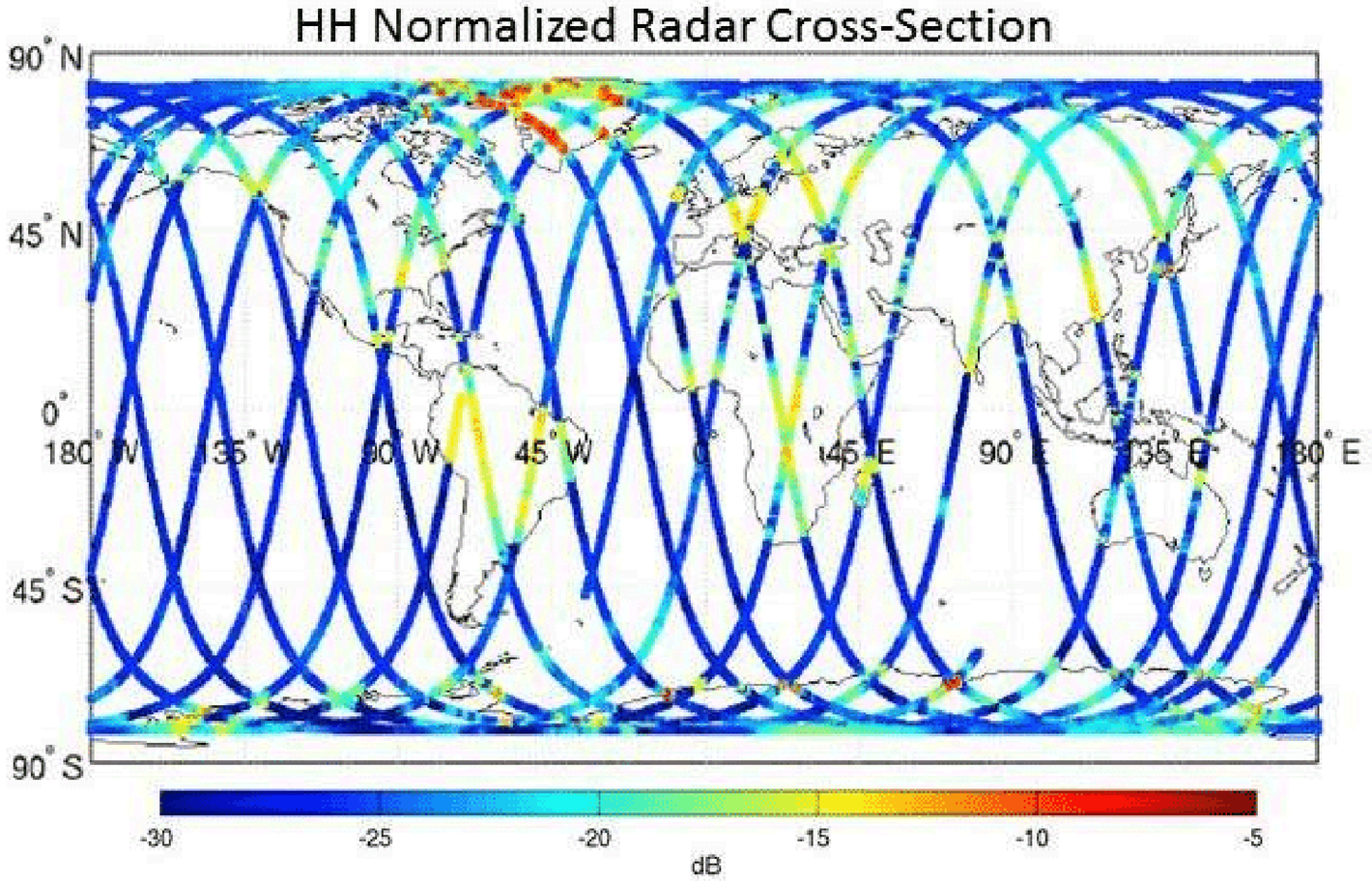
# Resolution

# Spatial Resolution



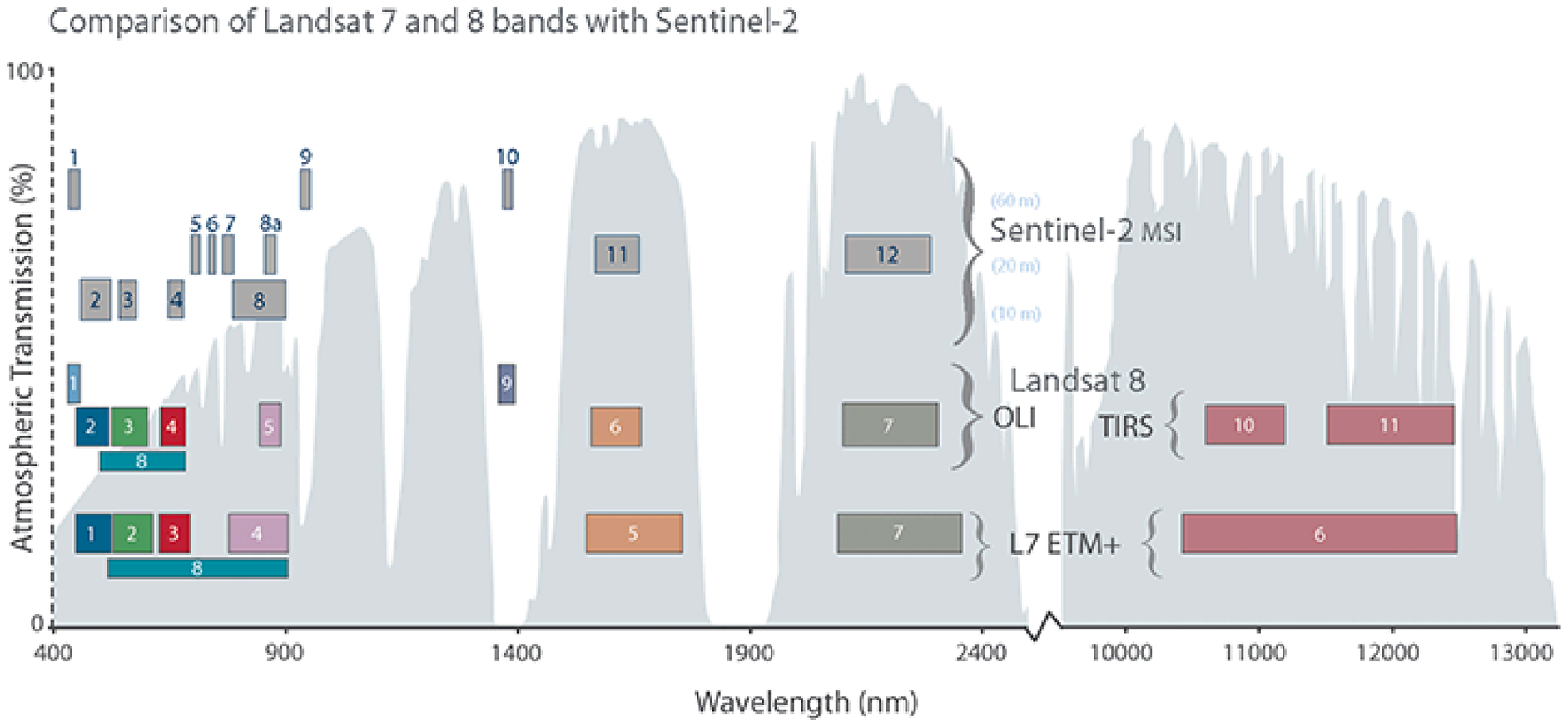
from left to right 1 m, 10 m, and 30 m

# Temporal Resolution




areas at high latitudes are imaged more frequently than the equatorial zone

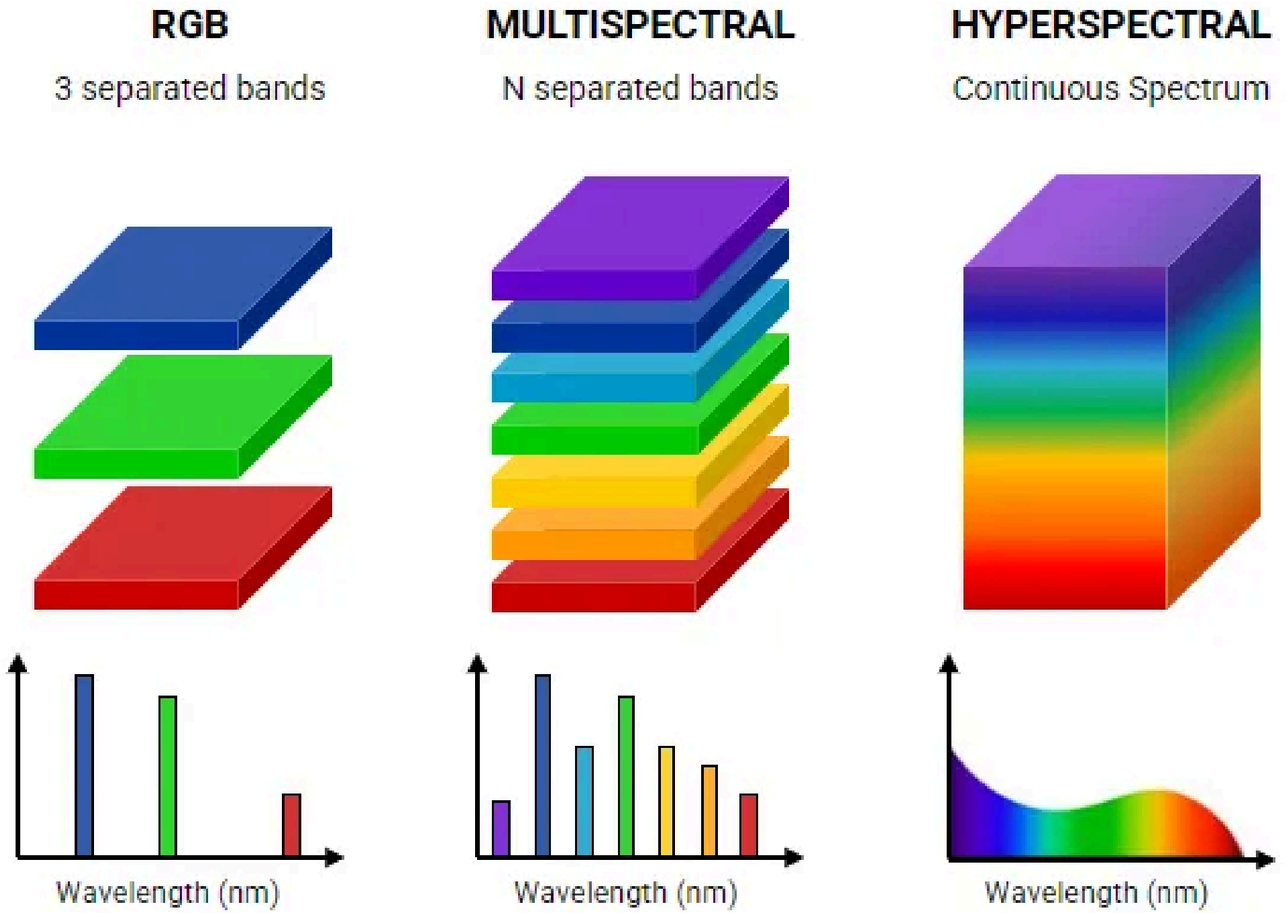
# Spectral Resolution



# Data

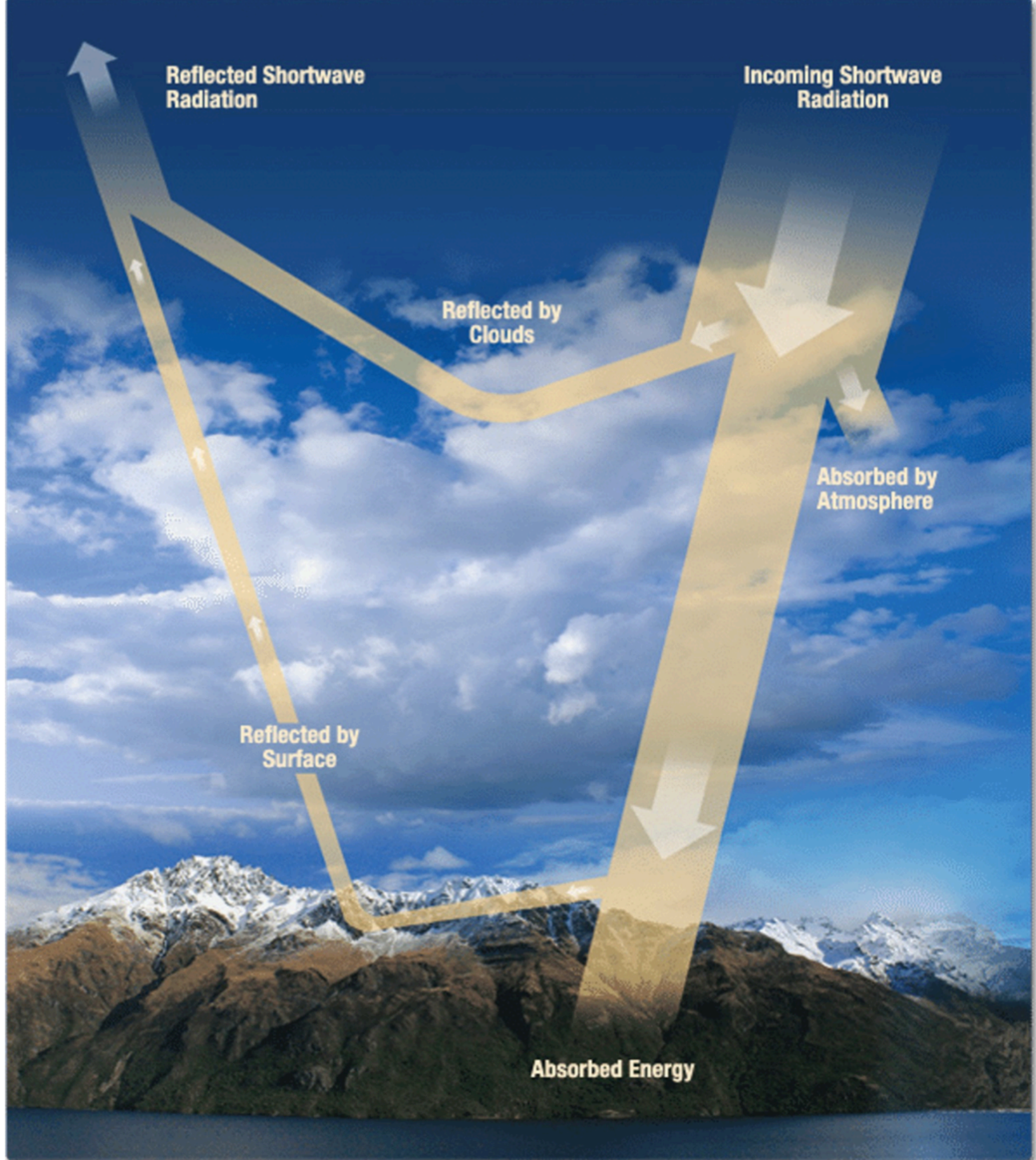
# Spectral Images

RGB (left) and multispectral (center) imaging only provide discrete and discontinuous portions of the spectral range.
Hyperspectral imaging (right) creates a hypercube, which yields the complete and continuous spectrum for each pixel of the image.



# Erors

**Internal errors** are created by the sensor itself.
**External errors** are due to platform perturbations, and the influence of the atmosphere.



# Data Preprocesing: Image Noise

**Striping/Banding**
Caused by sensor detector variations.
→ Use histogram matching to destripe the image.

**Line Drop**
Entire lines or sections contain faulty pixel values.
→ Replace with the average of pixels from the lines above and below.

**Bit Errors**
Appears as random "salt and pepper" noise.
→ Use a neighbourhood filter to detect and replace outliers with local averages.

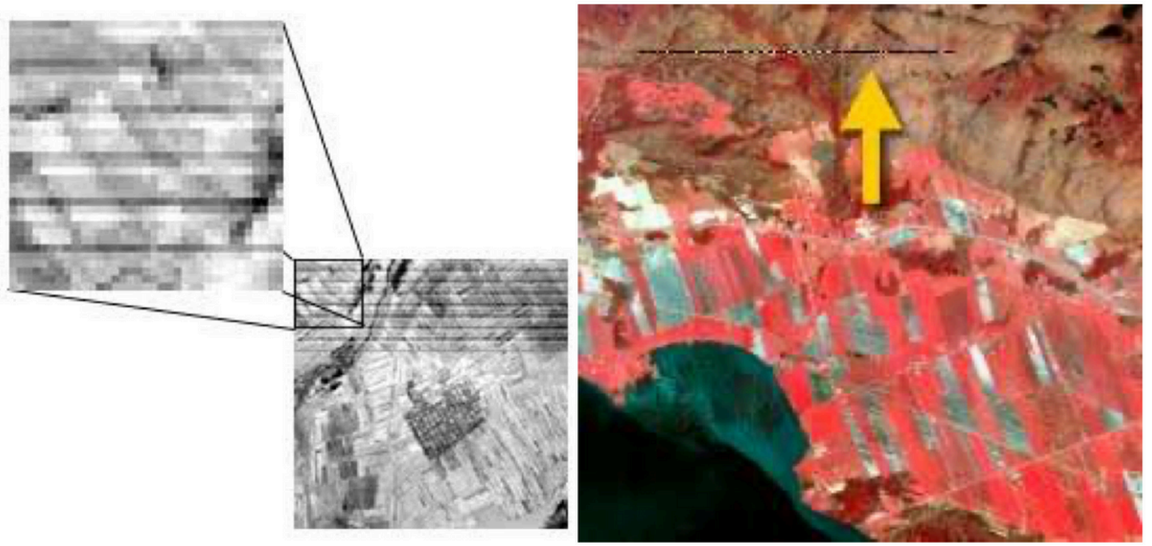


Figure 4.4: Examples of image noise showing (left) the striping effect for Landsat MSS and (right) dropped lines for Landsat TM (adapted from ccrs.nrcan.gc.ca).

# Data Preprocesing: Atmospheric Correction

**Rayleigh Scattering**
Caused by air molecules; stronger at shorter wavelengths. (Makes the sky appear blue.)
→ Apply atmospheric correction like the darkest pixel method.

**Mie Scattering**
Caused by aerosols (dust, smoke, etc.); depends on particle size and variability. Affects radiance and is hard to predict.
→ Use advanced correction models that include aerosol properties.

**Non-Selective Scattering**
Caused by large particles like rain or dust; not wavelength-dependent. Minor on clear days.
→ Usually negligible, but can be adjusted with full atmospheric correction

Figure 4.5: Example of Landsat TM image before (left) and after (right) atmospheric correction (from Liang *et al*., 2001)

# Data Preprocesing: Geometric Correction

**Image-to-Map Registration**
Aligns the image to real-world coordinates using Ground Control Points (GCPs) from maps (e.g., road intersections).

**Image-to-Image Registration**
Aligns one image to another (e.g., for time-series analysis).

**Resampling Methods**
After transformation, new pixel values are calculated using techniques like nearest neighbor, bilinear, or cubic convolution.

Figure 4.7: Example image before (left) and after (right) lines were shifted to correct for roll distortion of an airborne sensor acquisition (from Schott, 1996).

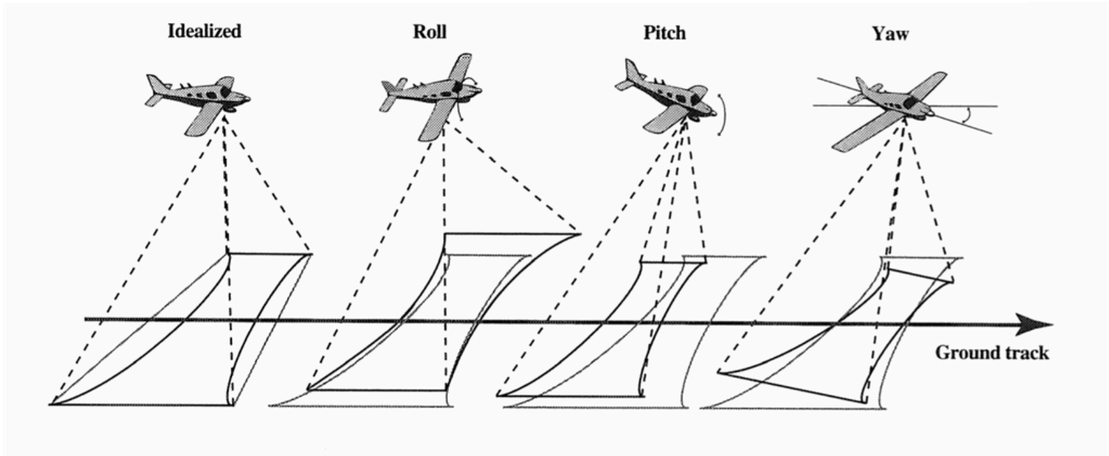
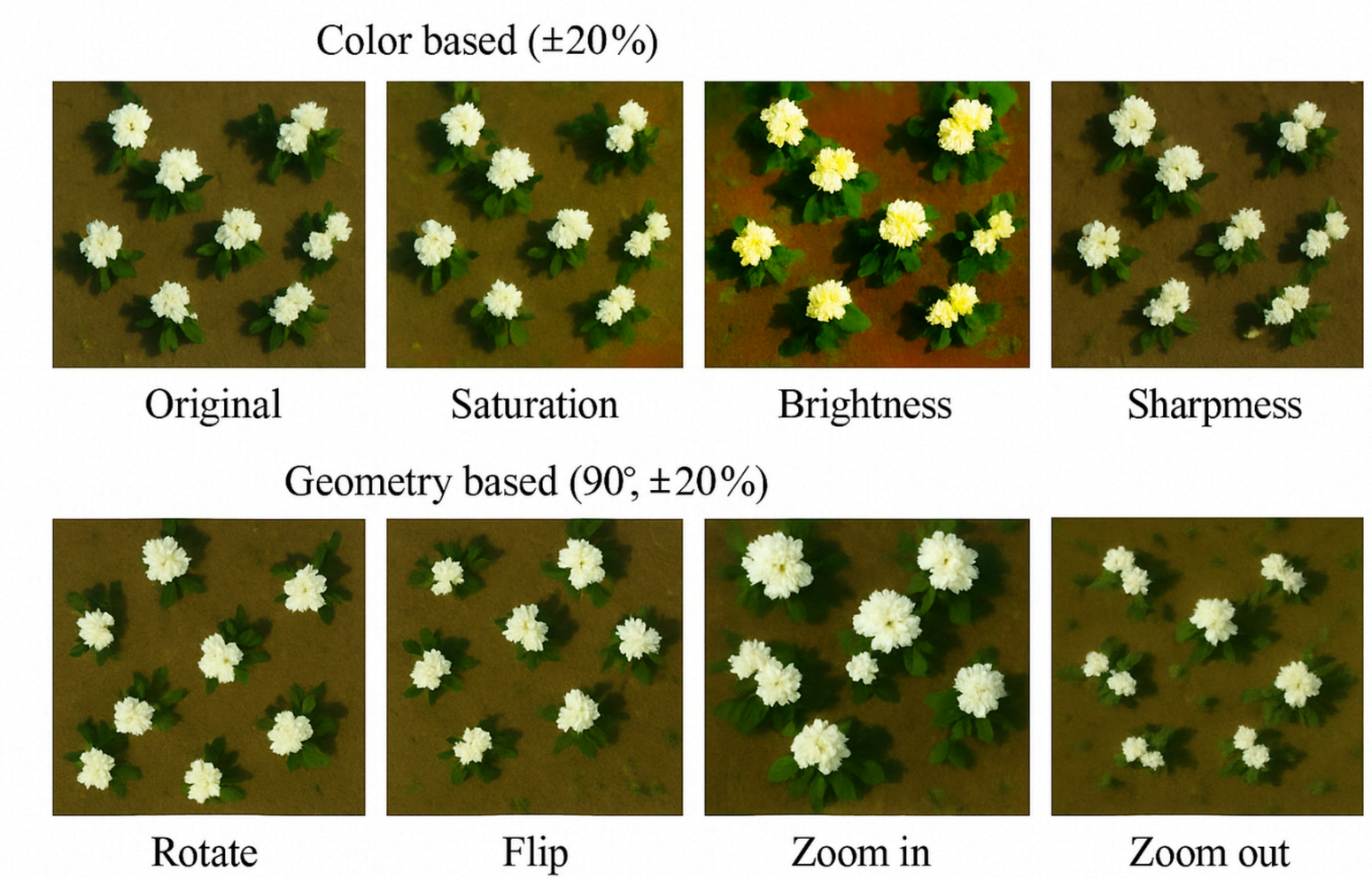



Figure 4.6: Geometric distortions due to aircraft orientation. Gray boundaries represent nominal coverage; black boundaries represent actual coverage (from Schott, 1996).

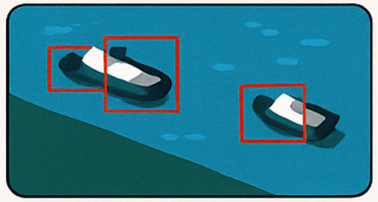
# Data Augmentation



# Applications of Deep Learning in Geospatial Data

# Applications of Deep Learning in Geospatial Data

## OBJECT DETECTION



Locating and labeling specific objects in images

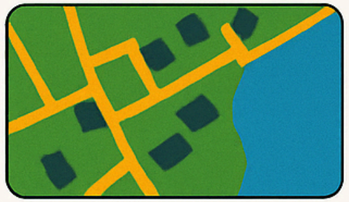
**Example:** Detecting ships in ports or vehicles in traffic surveillance using YOLO or Faster C-CNN

## TIME SERIES FORECASTING

Predicting changes in spatial data over time

**Example:** Foreecasting vegeation health (NDVI) using historical satellite time series with LSTM mode

## SEMANTIC SEGMENTATION



Pixel-wise classification to identify **spatial** structures.

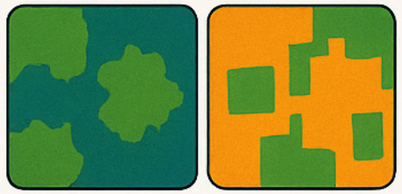
**Example:** Road and building footprint extraction from high-resolution satellite imagery

## IMAGE CLASSIFICATION

Assigning labels to **satellite or aerial images.**

**Example:** Land cover classification (e.g, forest, urban, water) from Sentinel-2 imagery using CNNs

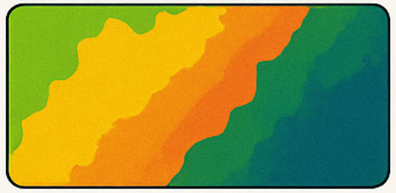
## CHANGE DETECTION



Identifying differences in geospatial data over time

**Example:** Monitoring defores:tation or urban expansion between two dates using Stamese networks

## REGRESSION

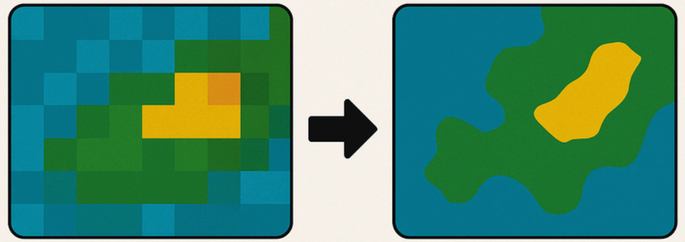


Predicting continuous spatial variables.

**Example:** Extimating air pollution levels (e.g, PM2.5 concentration based on sateliite data and weather

## SUPER-RESOLUTION & IMAGE ENHANCEMENT

Increasing spatial resolution of satellite imagery



Low-resolution High-resolution

**Example:** Increasing the resolution of Sentinel-2 imagery using a super-resolution model

# Basic Models

# Unet

# UNet: Revolutionizing Biomedical Image Segmentation

Developed in 2015 by Olaf Ronneberger's team to overcome the lack of annotated medical images. Before UNet, segmentation struggled with low accuracy, reliance on handcrafted features, and limited scalability.

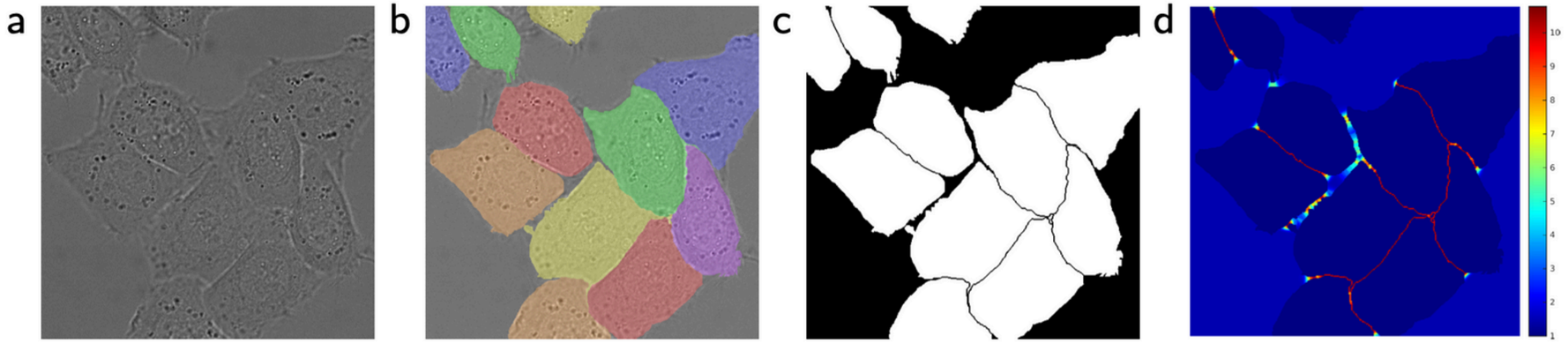
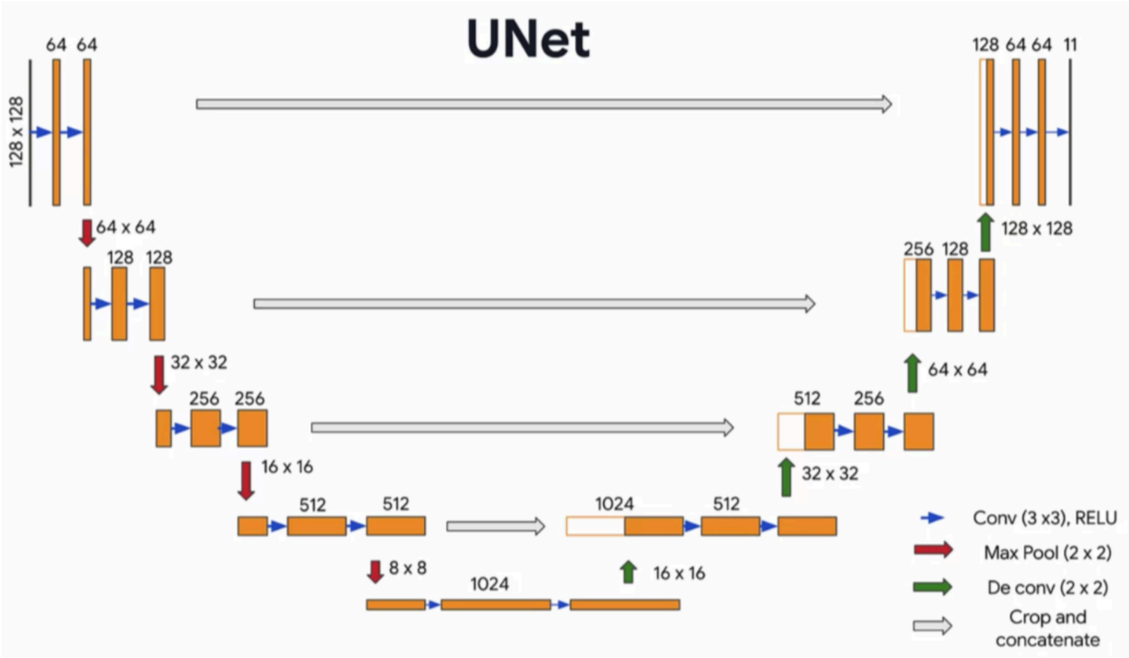



**Fig. 3.** HeLa cells on glass recorded with DIC (differential interference contrast) microscopy. (**a**) raw image. (**b**) overlay with ground truth segmentation. Different colors indicate different instances of the HeLa cells. (**c**) generated segmentation mask (white: foreground, black: background). (**d**) map with a pixel-wise loss weight to force the network to learn the border pixels.

# UNet Innovations

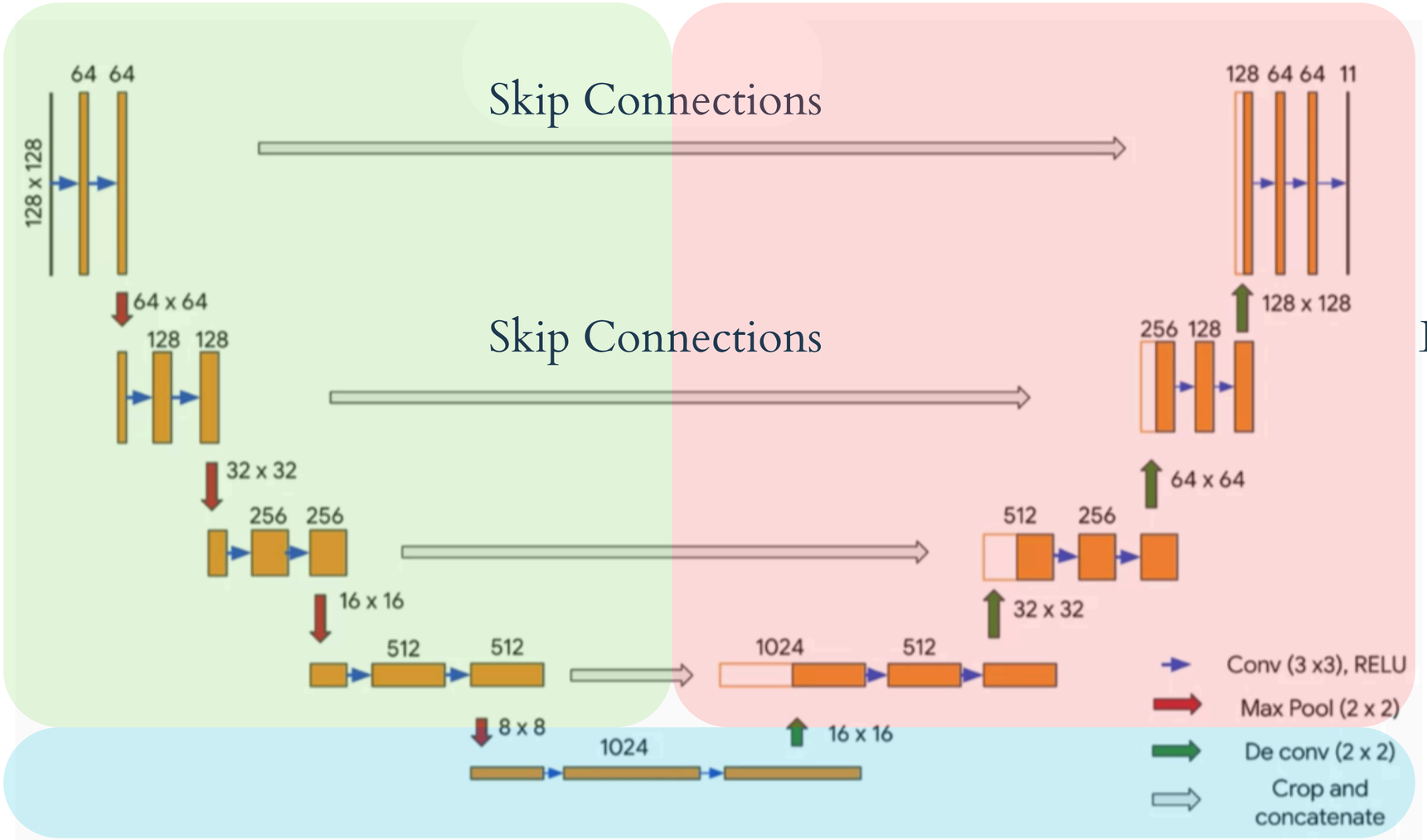
UNet introduced three key innovations:

- Symmetric expanding path for precise reconstruction.
- Skip connections to retain spatial context.
- Data augmentation to boost performance on small datasets.



# The U-Net architecture

Contracting Path (Encoder)

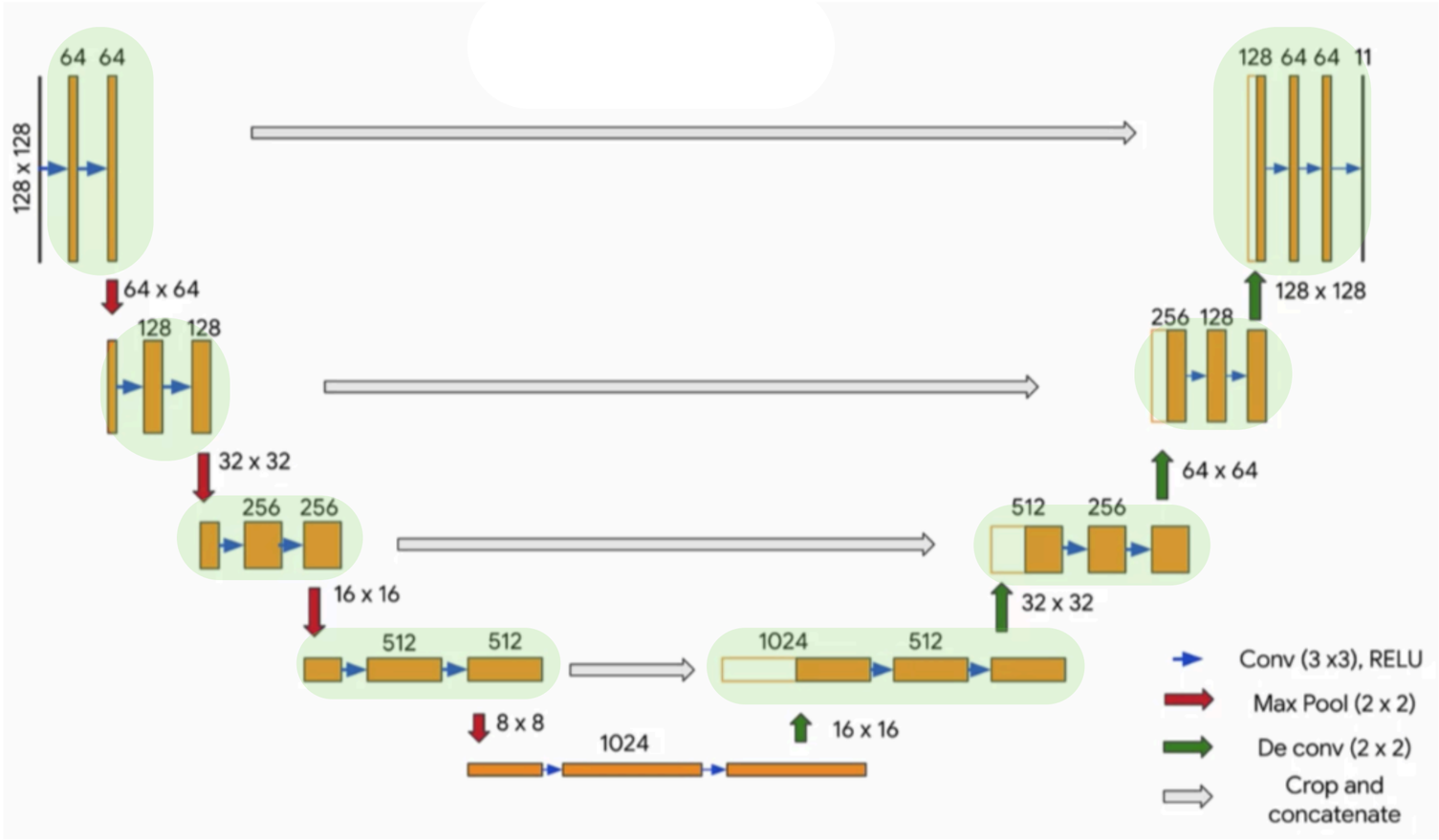



Expansive Path (Decoder)

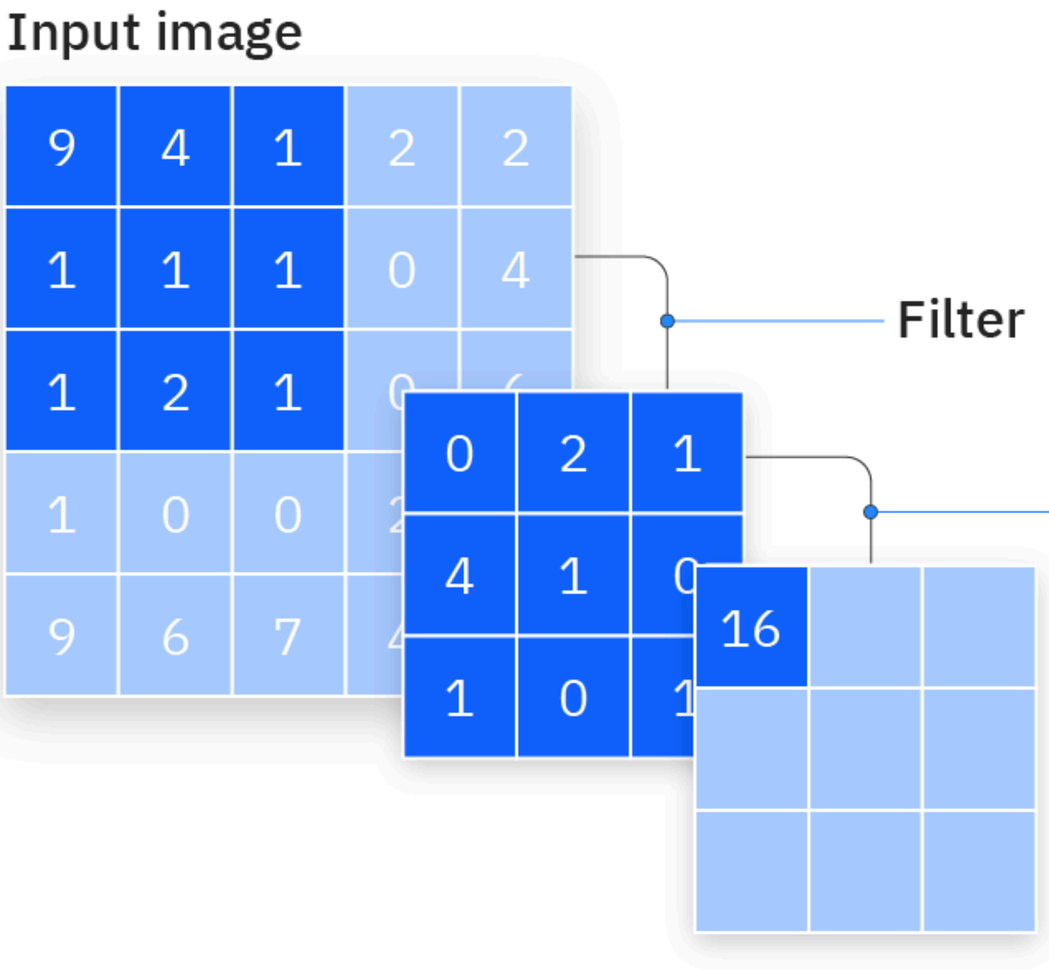
Bottleneck

# Encoder

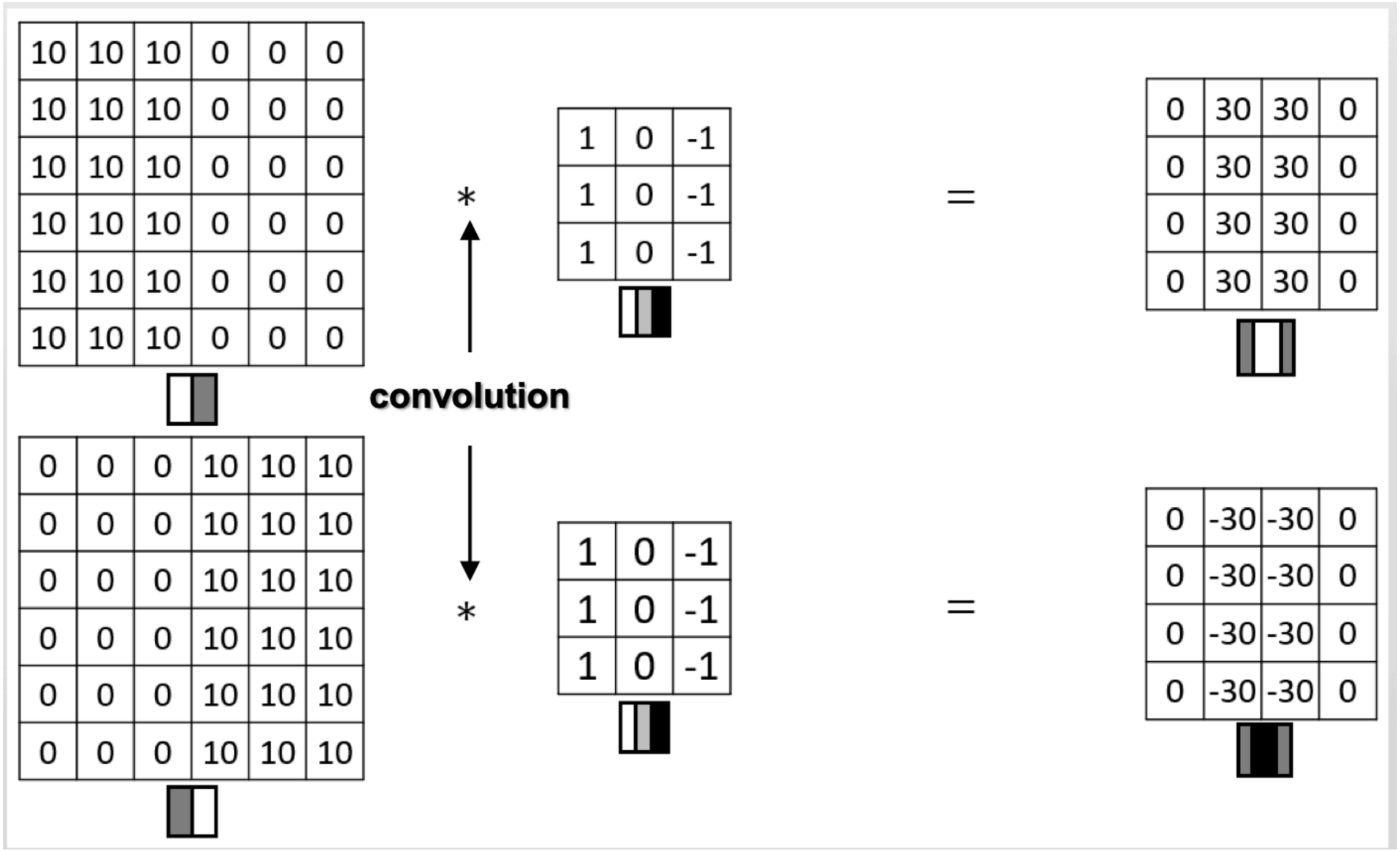
- **Convolutional Layers**
- Activation Functions
- Max Pooling
- Feature Doubling



# Encoder: Convolutional Layers



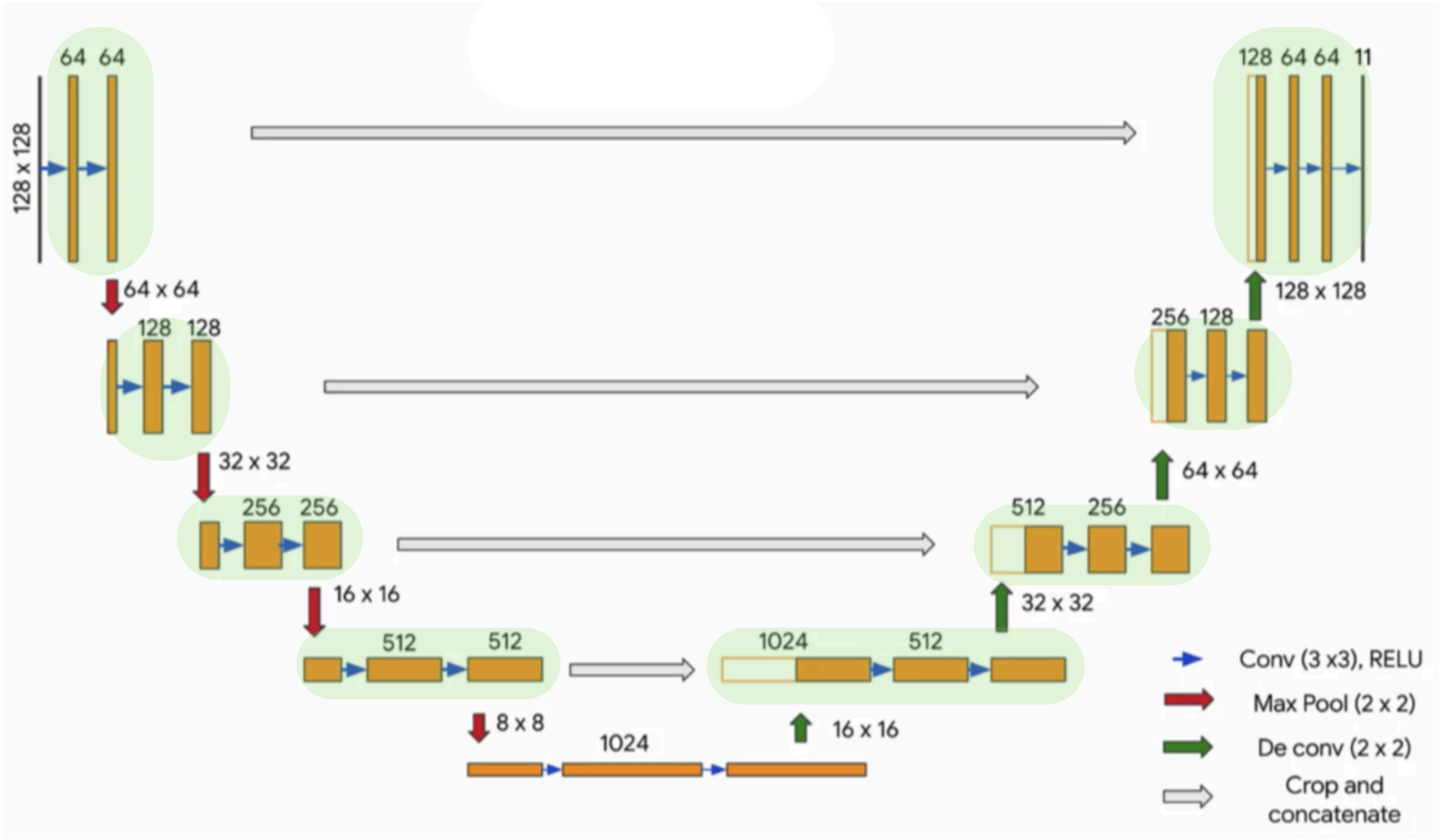

Output [0][0] = (9*0) + (4*2) + (1*4)
+ (1*1) + (1* 0) + (1*1) + (2* 0) + (1*1)
= 0 + 8 + 1 + 4 + 1 + 0 + 1 + 0 + 1
= 16



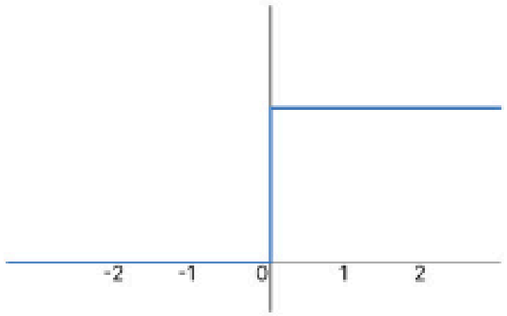

Vertical Line Detection Filter

# Encoder

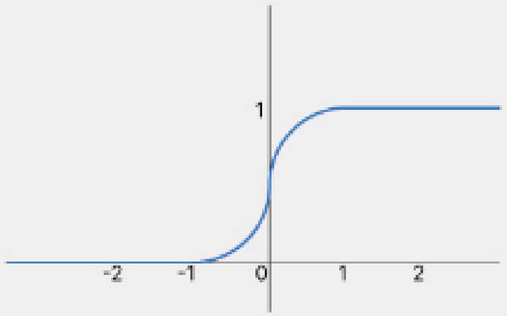
- Convolutional Layers
- **Activation Functions**
- Max Pooling
- Feature Doubling



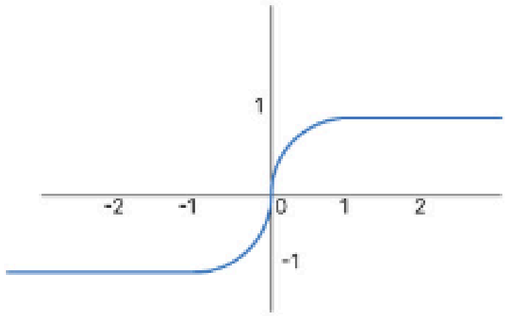
# Encoder: Activation Functions



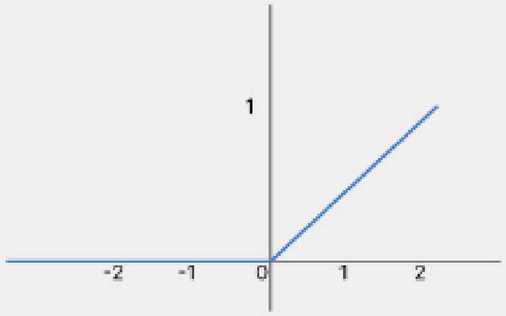

Step Function



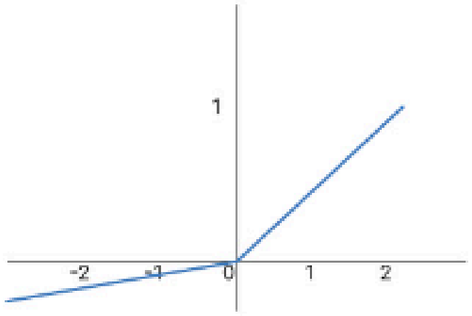

Sigmoid



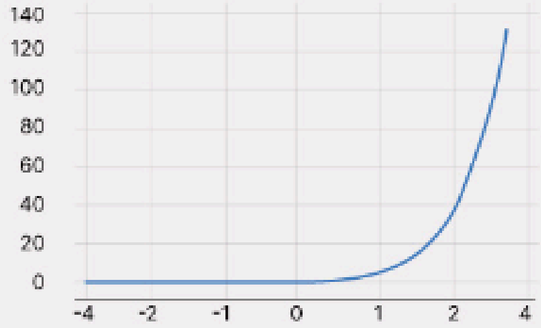

tanh




ReLU



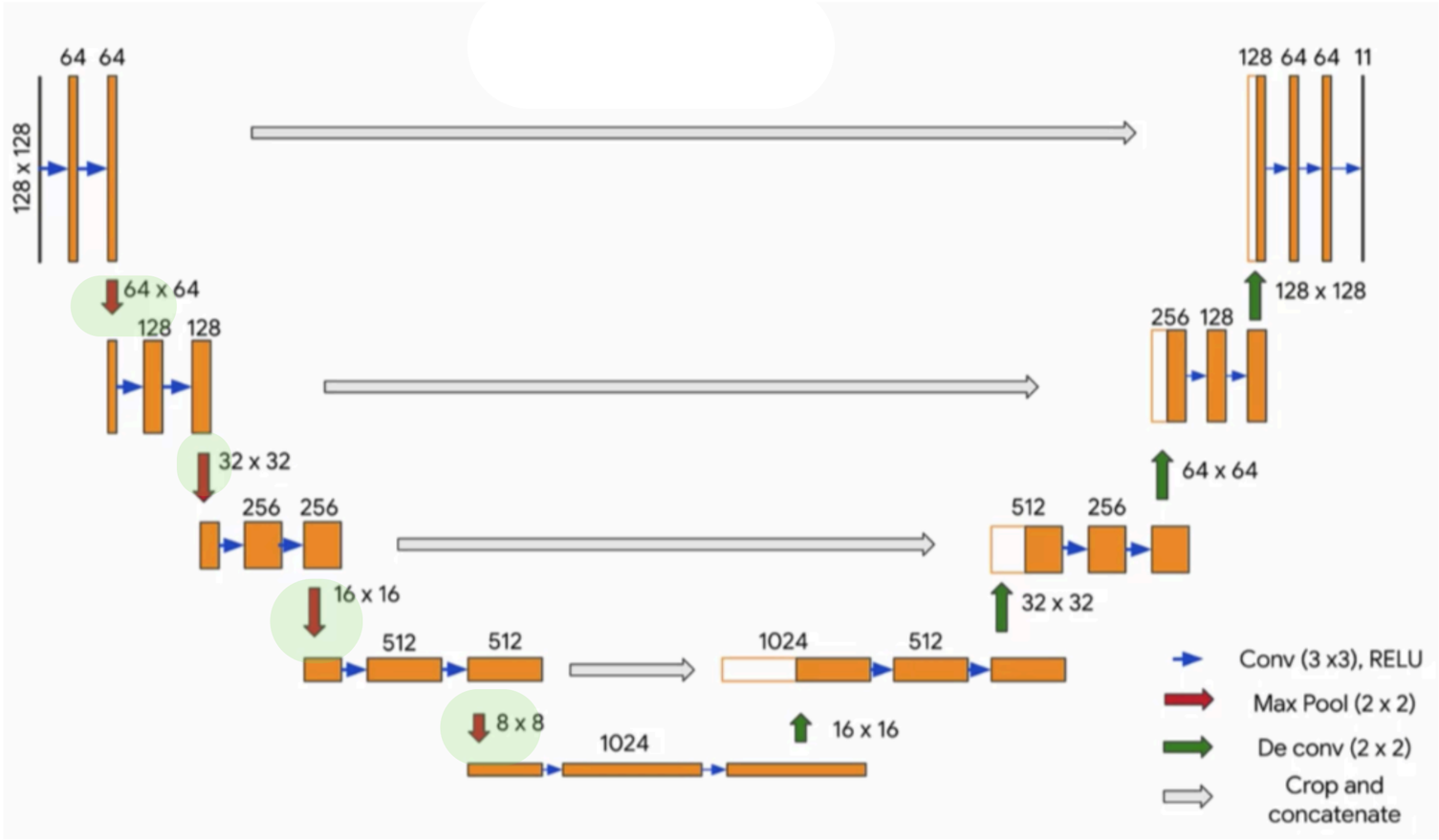

Leaky ReLU




Softmax Function

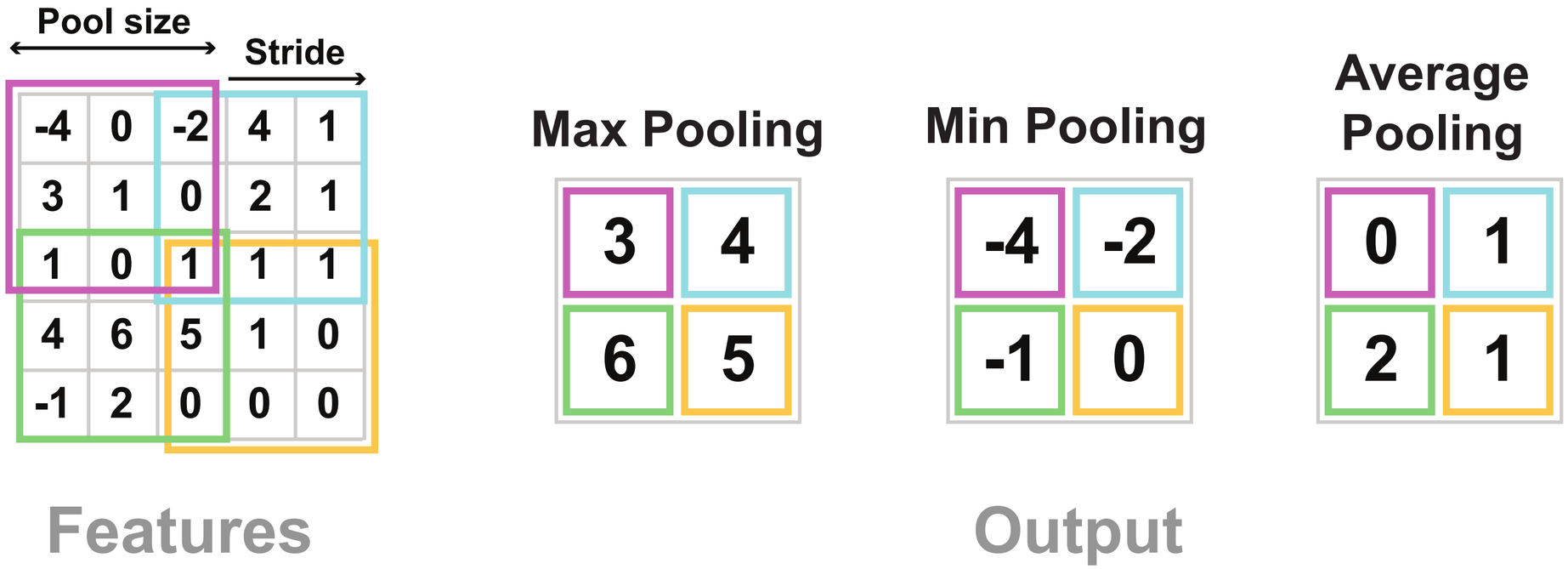
# Encoder

- Convolutional Layers
- Activation Functions
- **Max Pooling**
- Feature Doubling



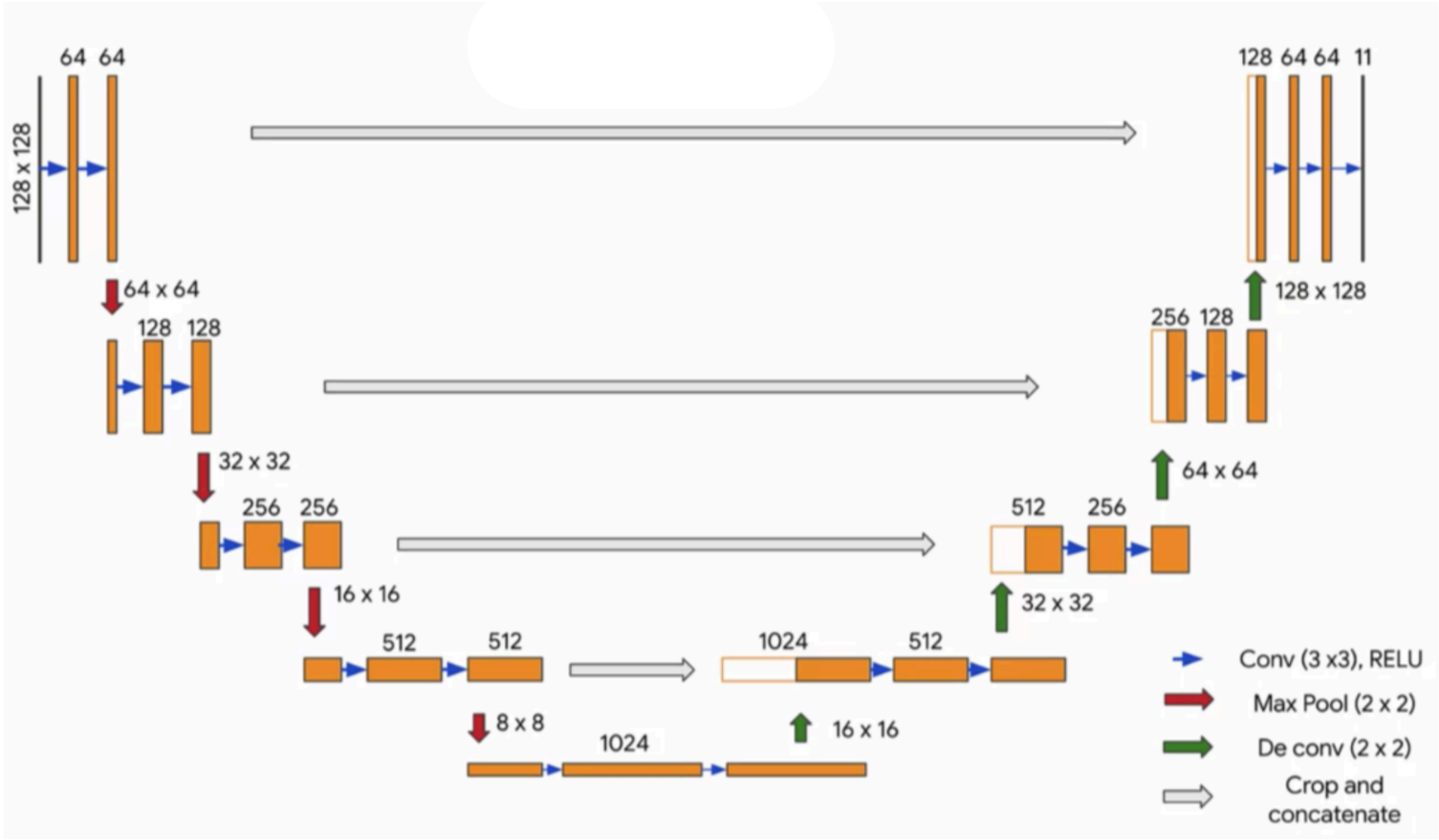
# Encoder: Pooling

**Max pooling** selects the highest value within a region — emphasizing strong activations.
**Average pooling** computes the mean value — smoothing the feature map.
**Min pooling** selects the lowest value — highlighting areas of low intensity or contrast, which can be useful in specific tasks like edge detection or anomaly detection.



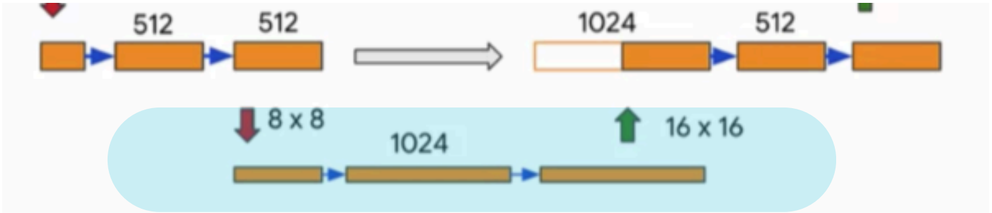
# Encoder

- Convolutional Layers
- Activation Functions
- Max Pooling
- **Feature Doubling**



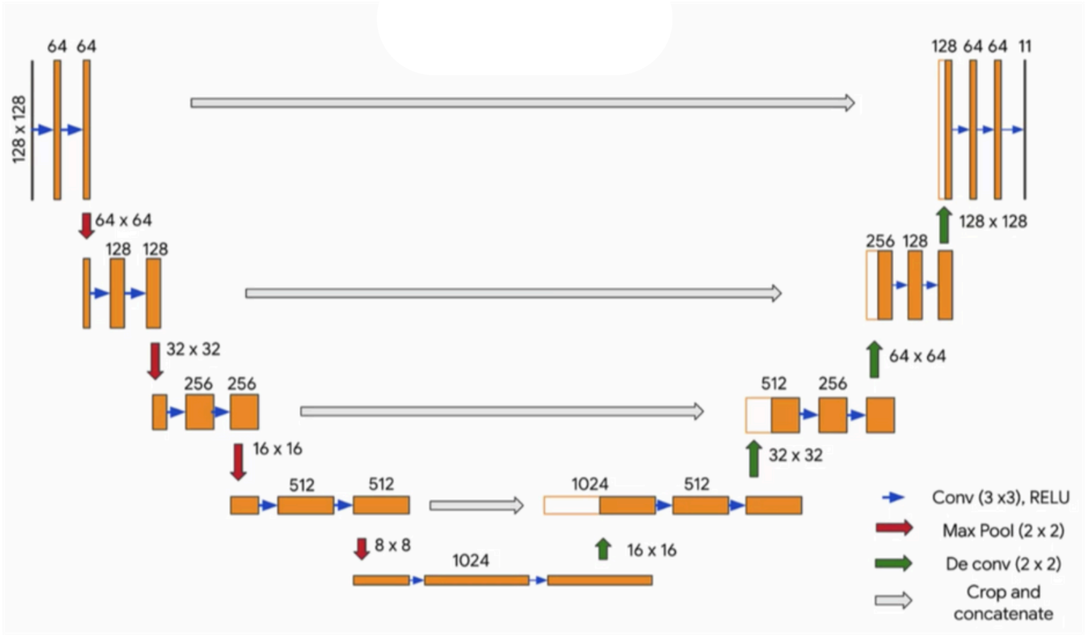
# Bottleneck

At the center of the U-Net is a bottleneck layer that captures the most critical features while maintaining spatial information.



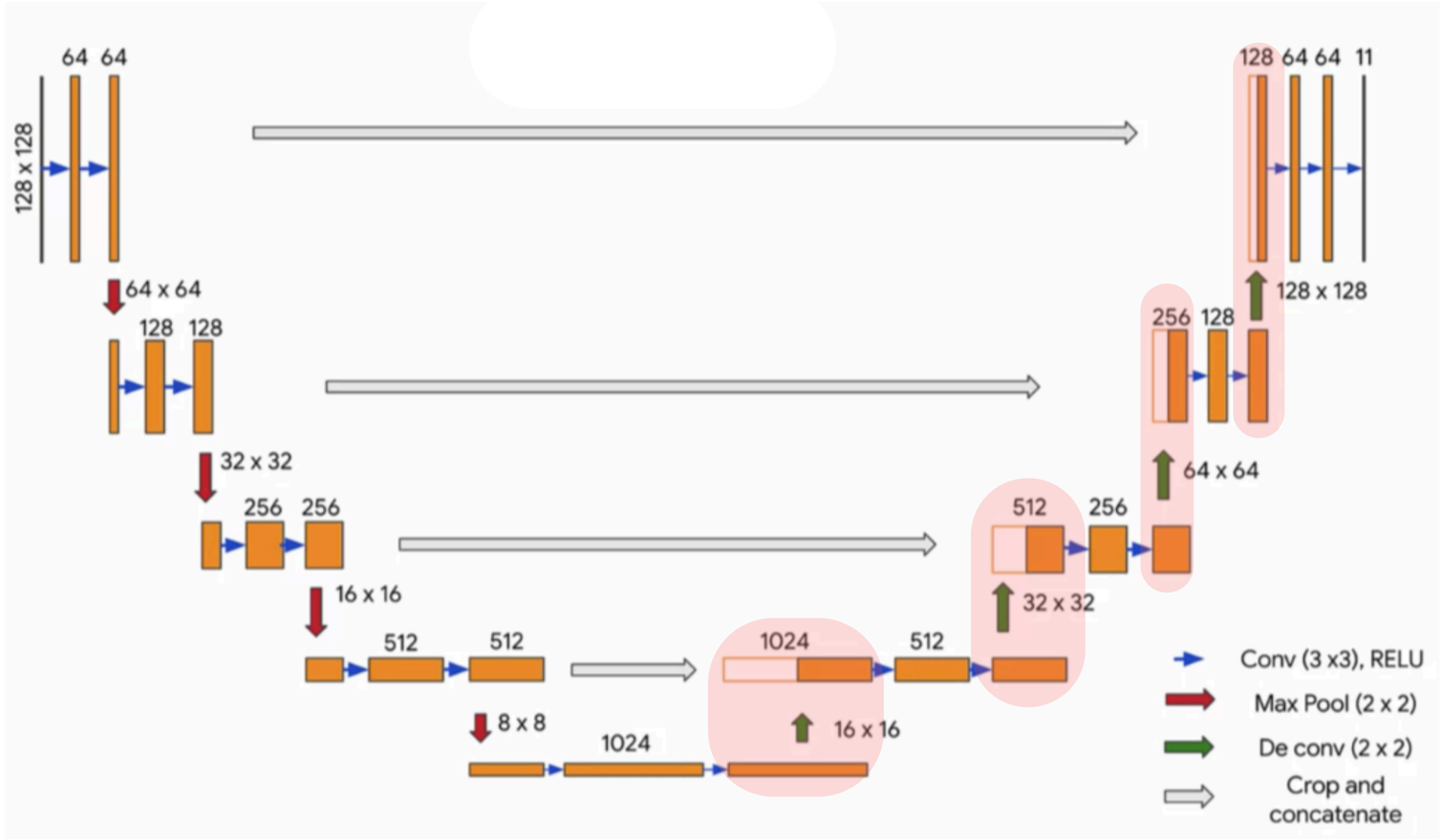
# Skip Connections

Skip connections link feature maps from the encoder to corresponding layers in the decoder.
They help recover fine spatial details lost during downsampling by directly passing high-resolution features.
This allows U-Net to combine context (from deep layers) with precision (from shallow layers) — improving segmentation accuracy, especially at object boundaries.

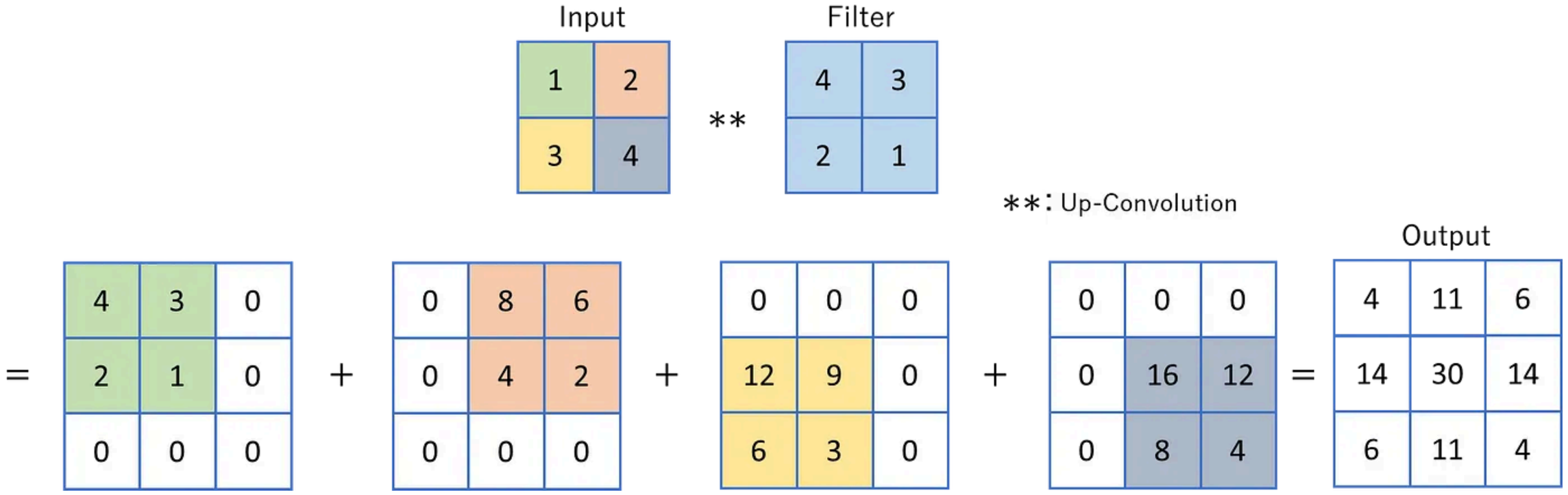


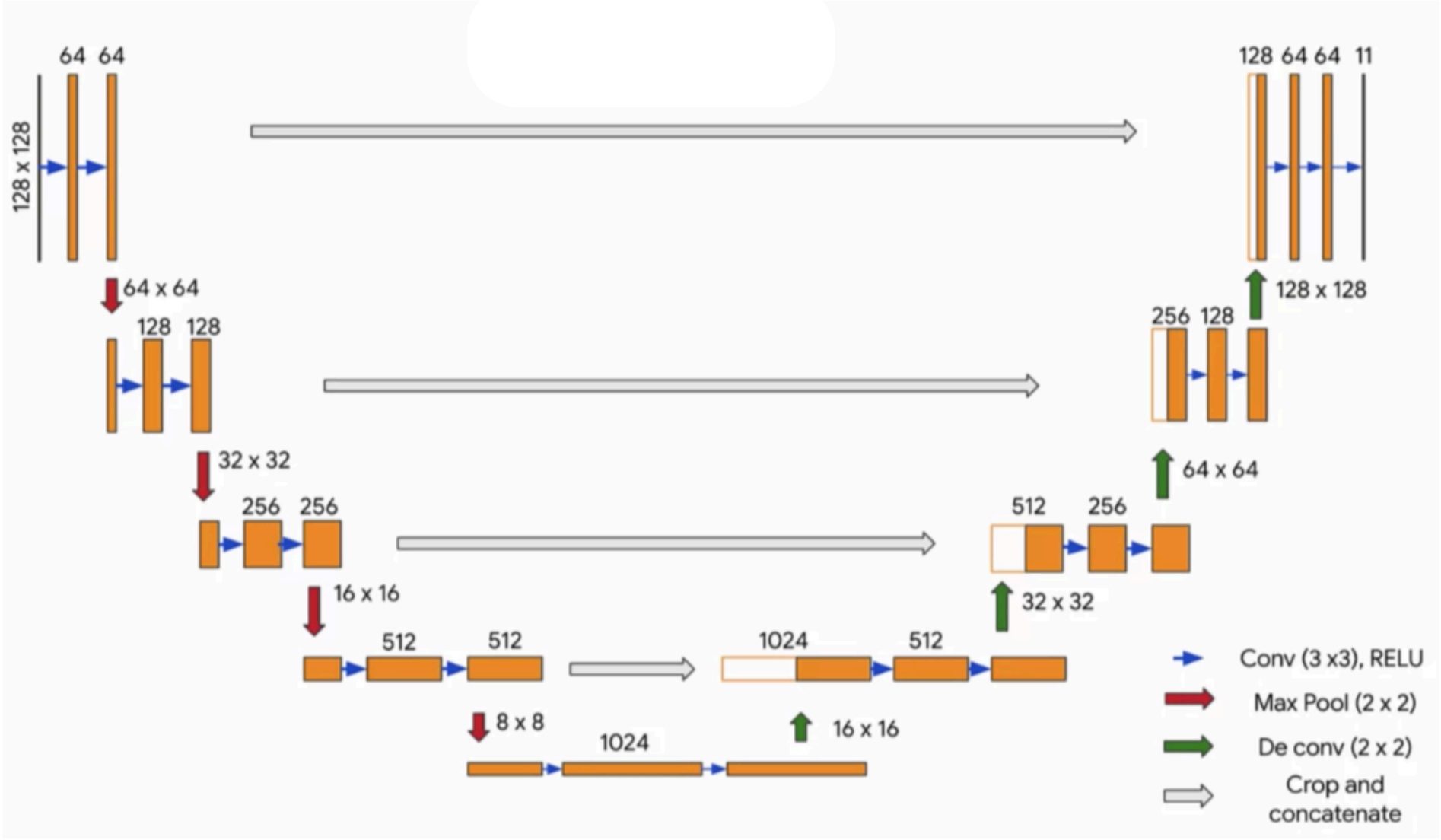
# Decoder

- **De convolution**
- Concatenation
- Convolutional Layers



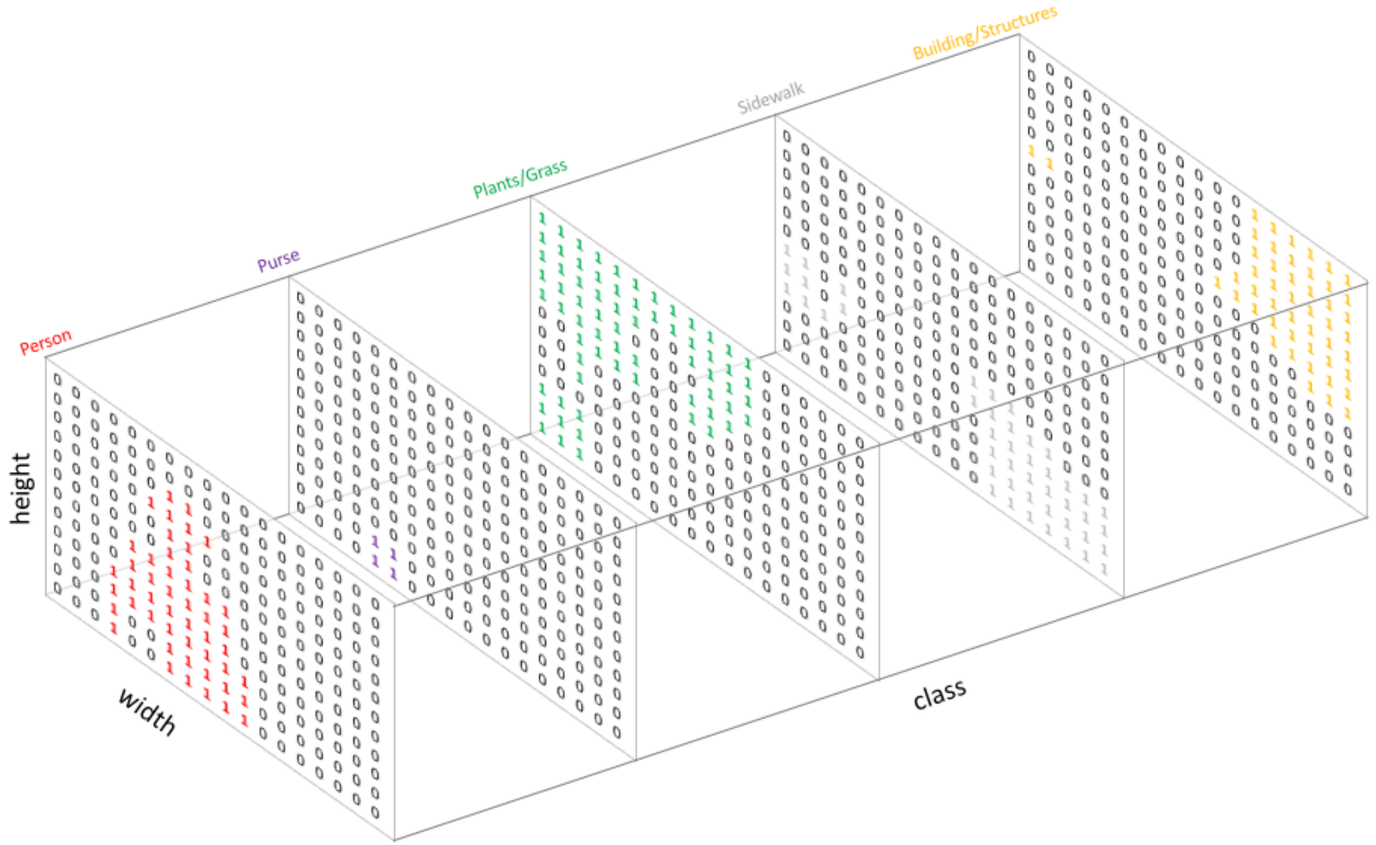
# Decoder: De Convolution




64 64
128 x 128
128 64 64 11
64 x 64
128 128
128 x 128
256 128
32 x 32
256 256
64 x 64
512 256
16 x 16
32 x 32
512 512
1024 512
8 x 8
16 x 16
1024
Conv (3 x3), RELU
Max Pool (2 x 2)
De conv (2 x 2)
Crop and concatenate

# Output

The network's output consists of as many layers as there are segmentation classes. Each layer indicates the likelihood of belonging to a specific class.



# Vision Transformers

# AN IMAGE IS WORTH 16X16 WORDS: TRANSFORMERS FOR IMAGE RECOGNITION AT SCALE

**Alexey Dosovitskiy**[*,†], **Lucas Beyer**[*], **Alexander Kolesnikov**[*], **Dirk Weissenborn**[*], **Xiaohua Zhai**[*], **Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, Neil Houlsby**[*,†]

[*]equal technical contribution, [†]equal advising

Google Research, Brain Team

{adosovitskiy, neilhoulsby}@google.com



## ABSTRACT

While the Transformer architecture has become the de-facto standard for natural language processing tasks, its applications to computer vision remain limited. In vision, attention is either applied in conjunction with convolutional networks, or used to replace certain components of convolutional networks while keeping their overall structure in place. We show that this reliance on CNNs is not necessary and a pure transformer applied directly to sequences of image patches can perform very well on image classification tasks. When pre-trained on large amounts of data and transferred to multiple mid-sized or small image recognition benchmarks (ImageNet, CIFAR-100, VTAB, etc.), Vision Transformer (ViT) attains excellent results compared to state-of-the-art convolutional networks while requiring substantially fewer computational resources to train.[1]

## 1 INTRODUCTION

Self-attention-based architectures, in particular Transformers (Vaswani et al., 2017), have become the model of choice in natural language processing (NLP). The dominant approach is to pre-train on a large text corpus and then fine-tune on a smaller task-specific dataset (Devlin et al., 2019). Thanks to Transformers' computational efficiency and scalability, it has become possible to train models of unprecedented size, with over 100B parameters (Brown et al., 2020; Lepikhin et al., 2020). With the models and datasets growing, there is still no sign of saturating performance.

In computer vision, however, convolutional architectures remain dominant (LeCun et al., 1989; Krizhevsky et al., 2012; He et al., 2016). Inspired by NLP successes, multiple works try combining CNN-like architectures with self-attention (Wang et al., 2018; Carion et al., 2020), some replacing the convolutions entirely (Ramachandran et al., 2019; Wang et al., 2020a). The latter models, while theoretically efficient, have not yet been scaled effectively on modern hardware accelerators due to the use of specialized attention patterns. Therefore, in large-scale image recognition, classic ResNet-like architectures are still state of the art (Mahajan et al., 2018; Xie et al., 2020; Kolesnikov et al., 2020).

Inspired by the Transformer scaling successes in NLP, we experiment with applying a standard Transformer directly to images, with the fewest possible modifications. To do so, we split an image into patches and provide the sequence of linear embeddings of these patches as an input to a Transformer. Image patches are treated the same way as tokens (words) in an NLP application. We train the model on image classification in supervised fashion.

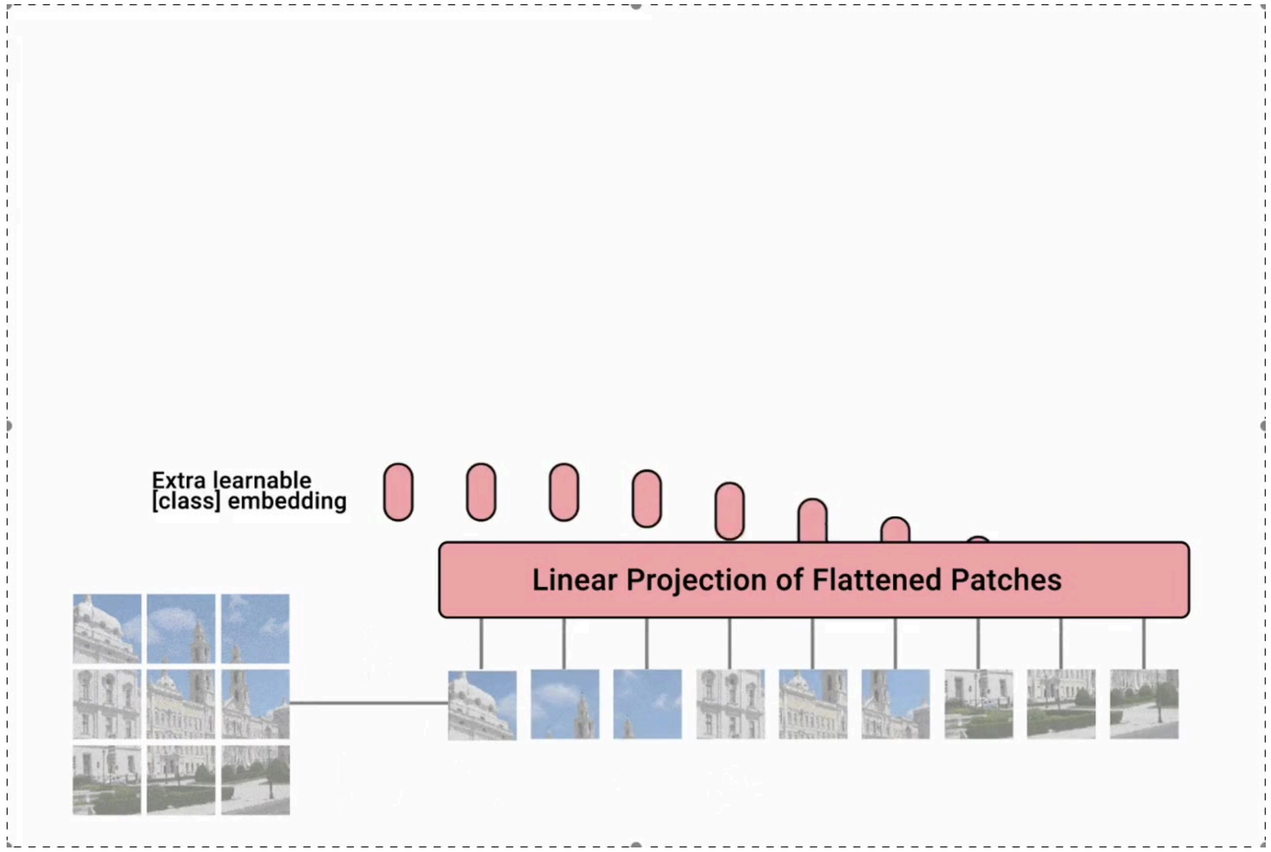
When trained on mid-sized datasets such as ImageNet without strong regularization, these mod-

# Vision transformer architecture

1. Split an image into patches
2. Flatten the patches
3. Produce lower-dimensional linear embeddings from the flattened patches
4. ...

# Vision transformer architecture

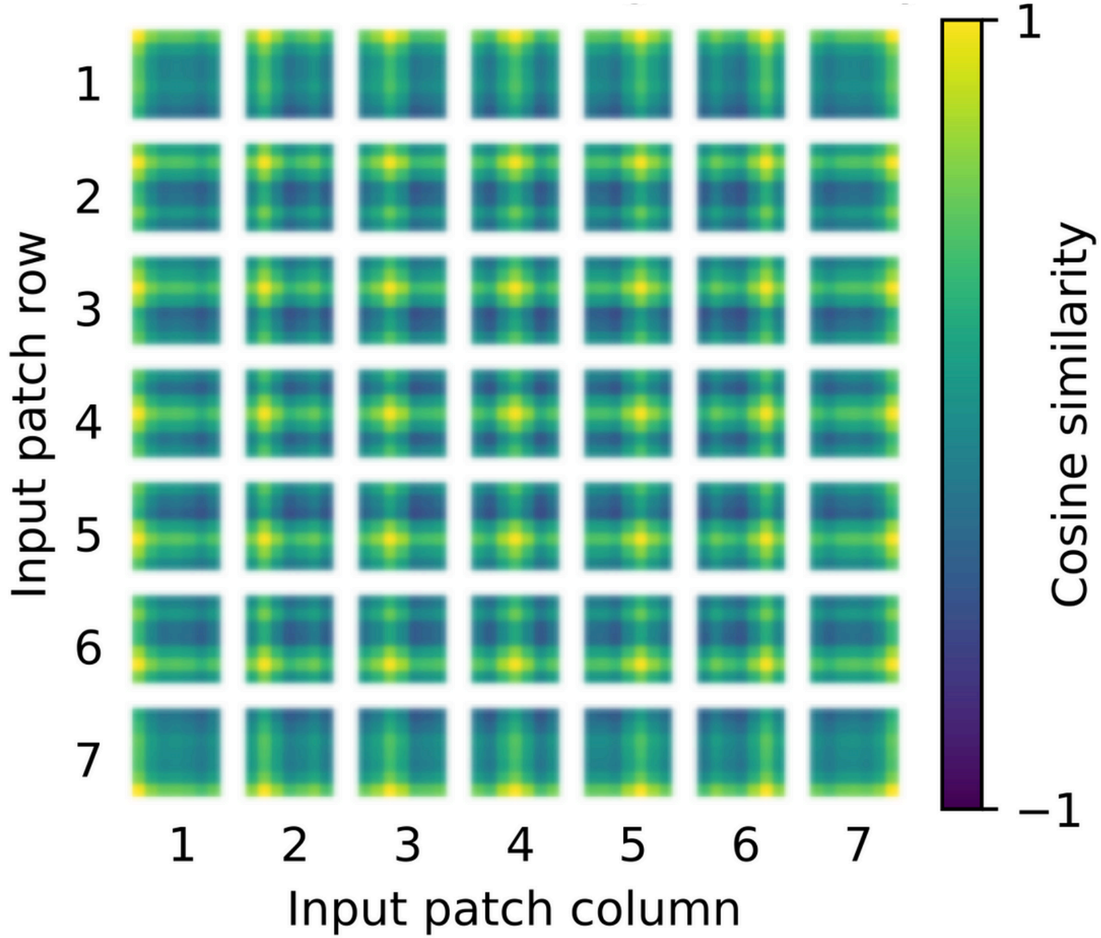
1. Split an image into patches
2. Flatten the patches
3. Produce lower-dimensional linear embeddings from the flattened patches
4. **Add positional embeddings**
5. ...

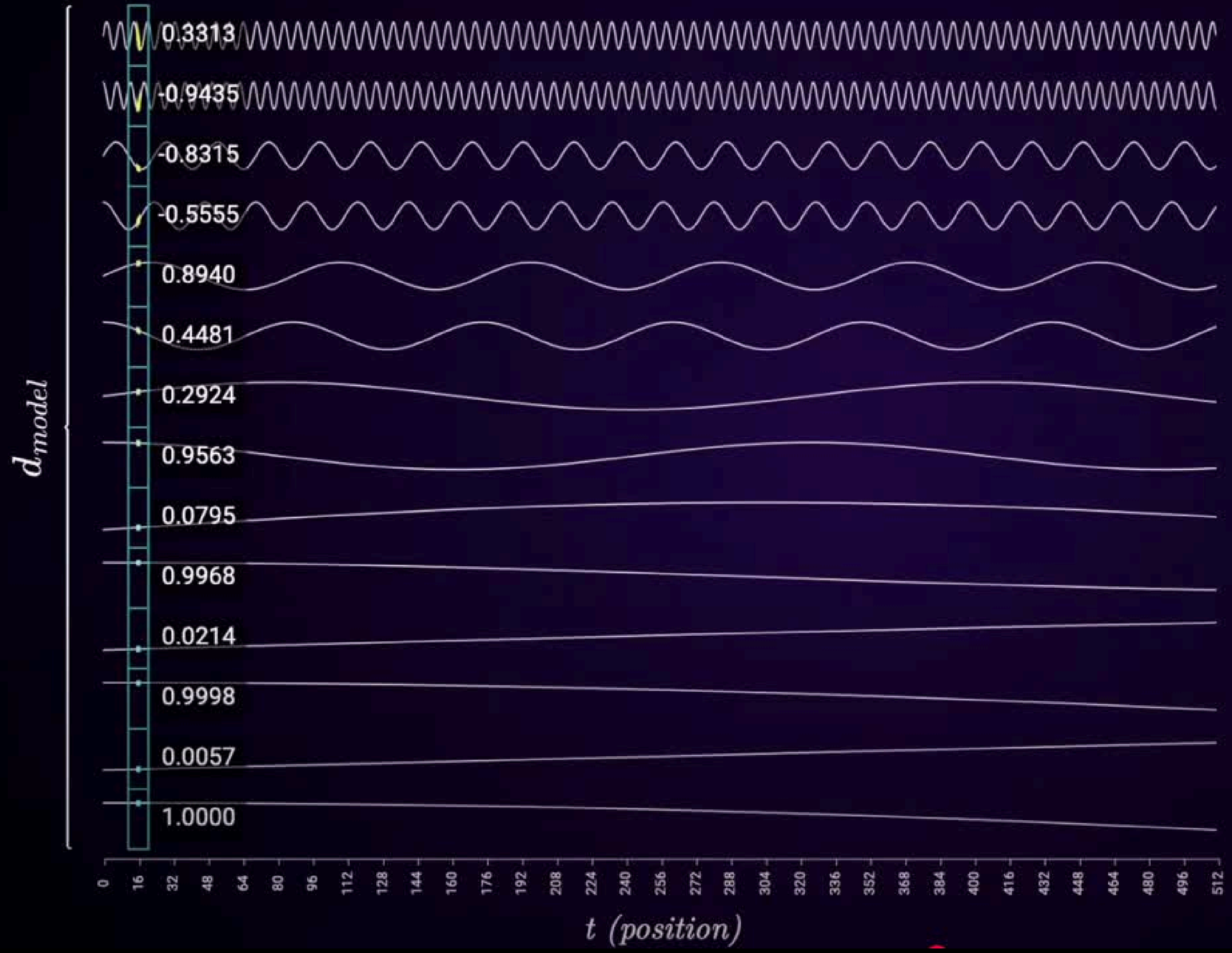


# Position embedding

$$PE(pos, 2_i) = sin(\frac{pos}{10000^{2i/d^{model}}})$$

$$PE(pos, 2_{i+1}) = cos(\frac{pos}{10000^{2i/d^{model}}})$$






*k is even:* $\sin\left(\frac{t}{N^{\frac{k}{d_{model}}}}\right)$

*k is odd:* $\cos\left(\frac{t}{N^{\frac{k}{d_{model}}}}\right)$

$L\ (context/sequence\ length) = 512$

$t\ (position) = \{1, \ldots, L\}$

$N = 10000$

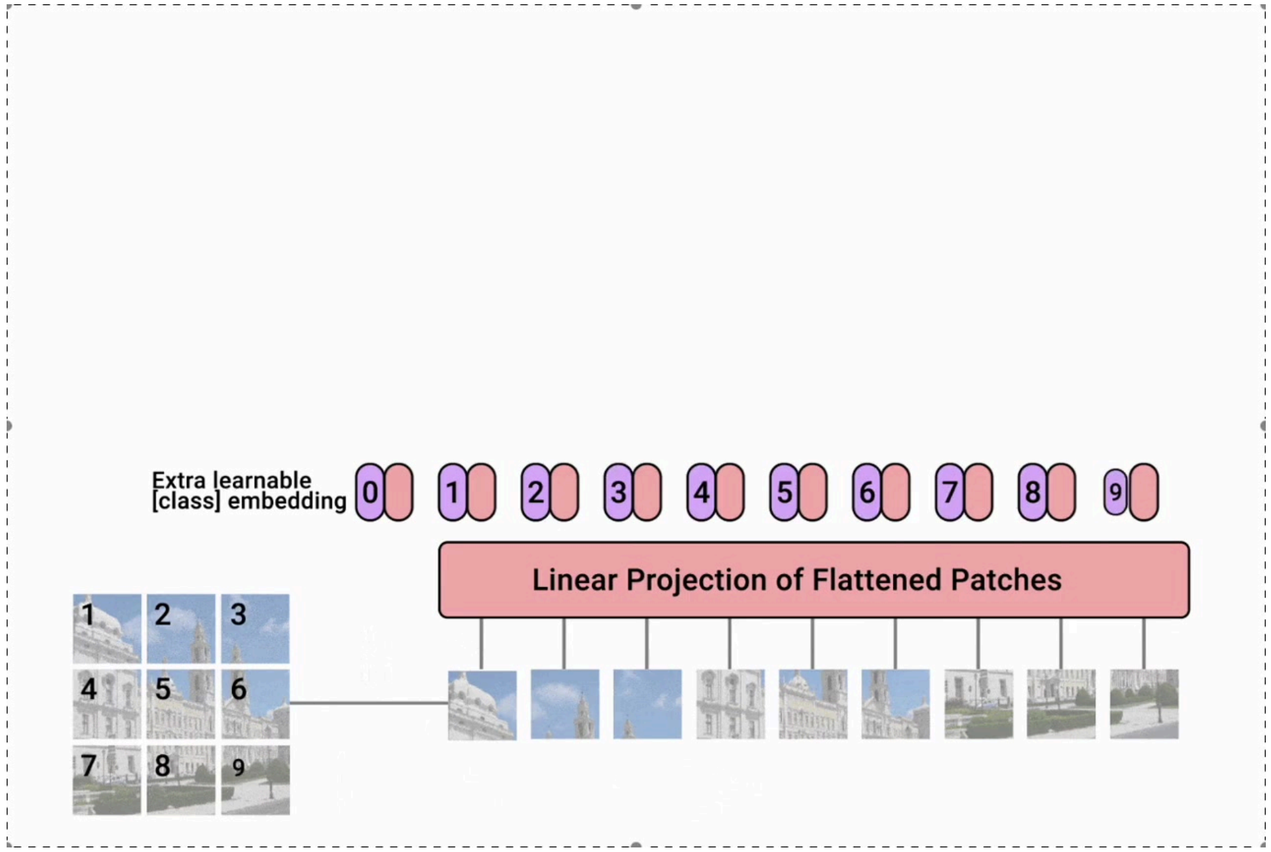
$d_{model} = 14$

$k = \{1, 2, \ldots, d_{model}\}$

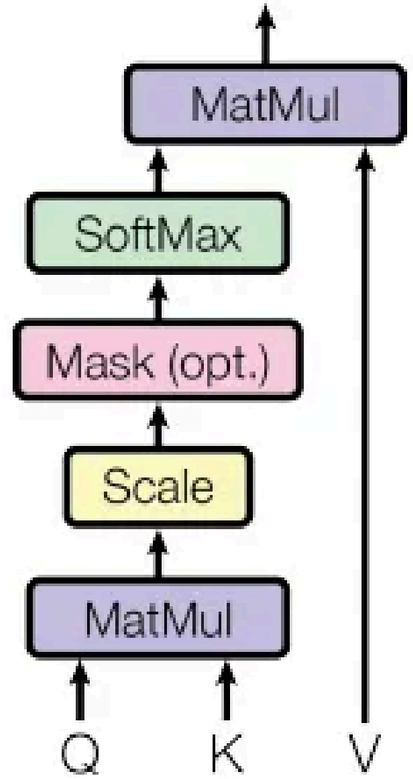

0 0 0 0 0 0 0 1
1

# Vision transformer architecture

1. Split an image into patches
2. Flatten the patches
3. Produce lower-dimensional linear embeddings from the flattened patches
4. Add positional embeddings
5. **Feed the sequence as an input to a standard transformer encoder**
6. ...

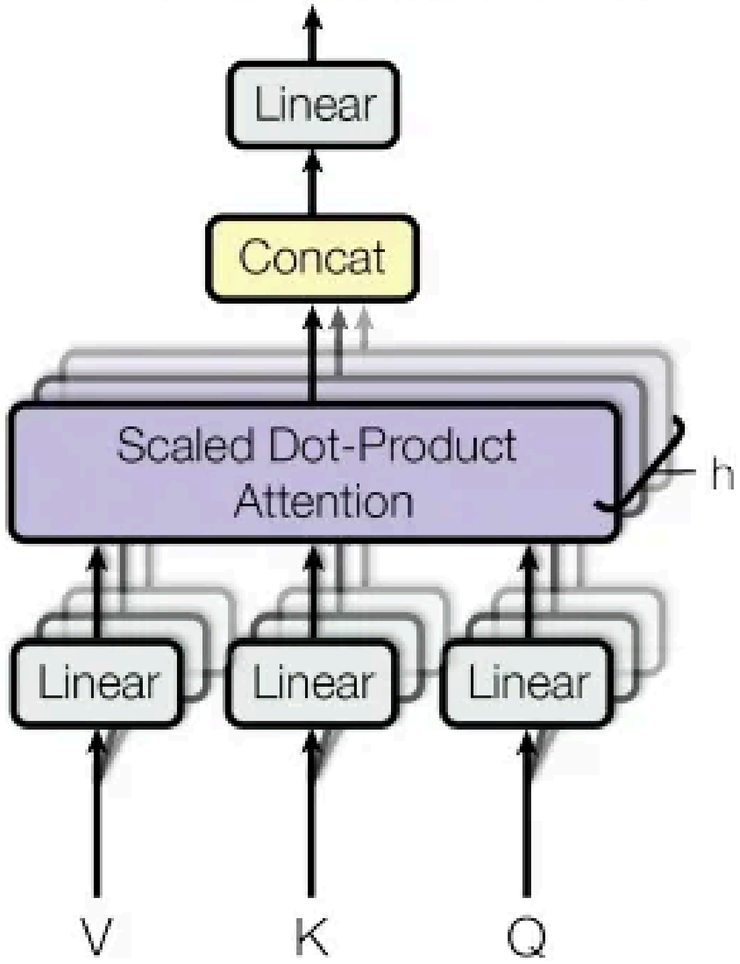


## Scaled Dot-Product Attention



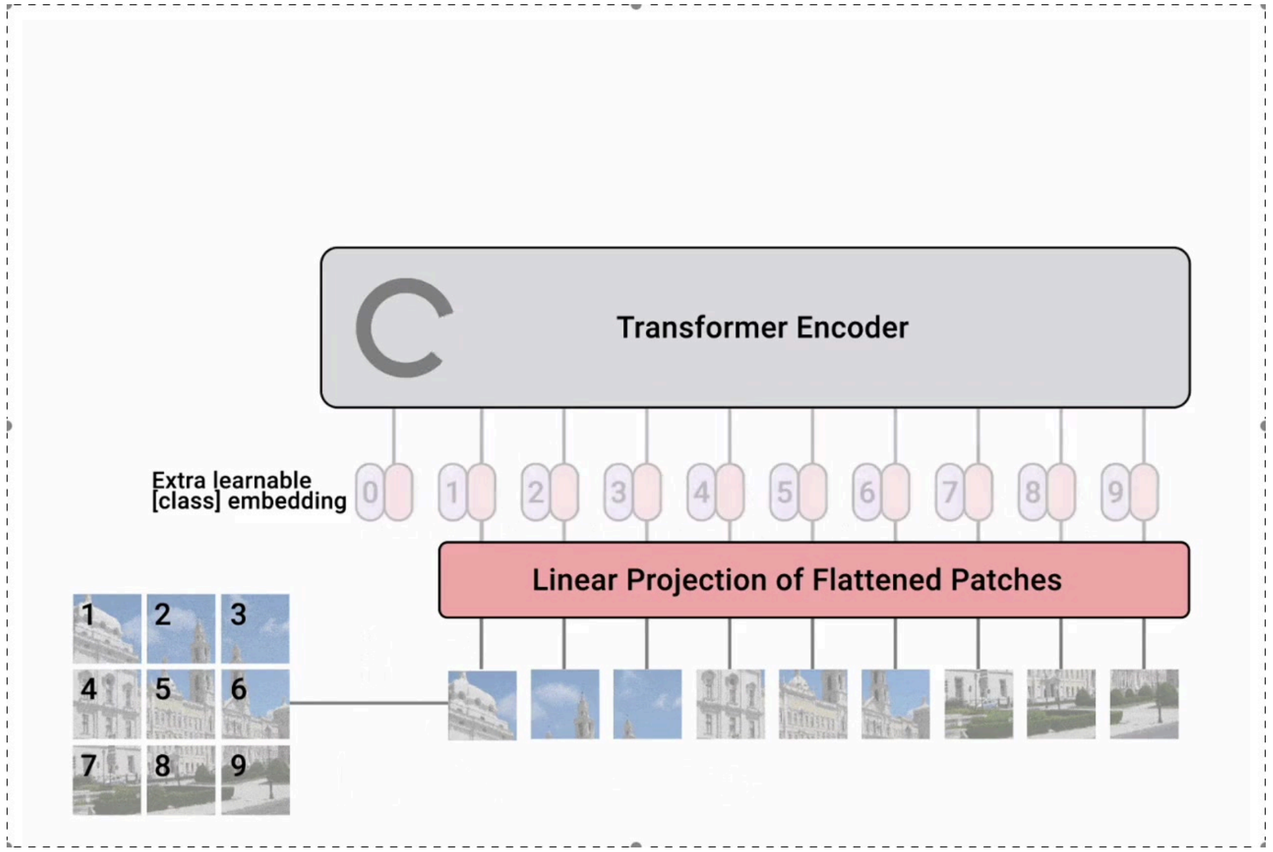

$$Attention = softmax(\frac{QK^T}{d_k})V$$

## Multi-Head Attention

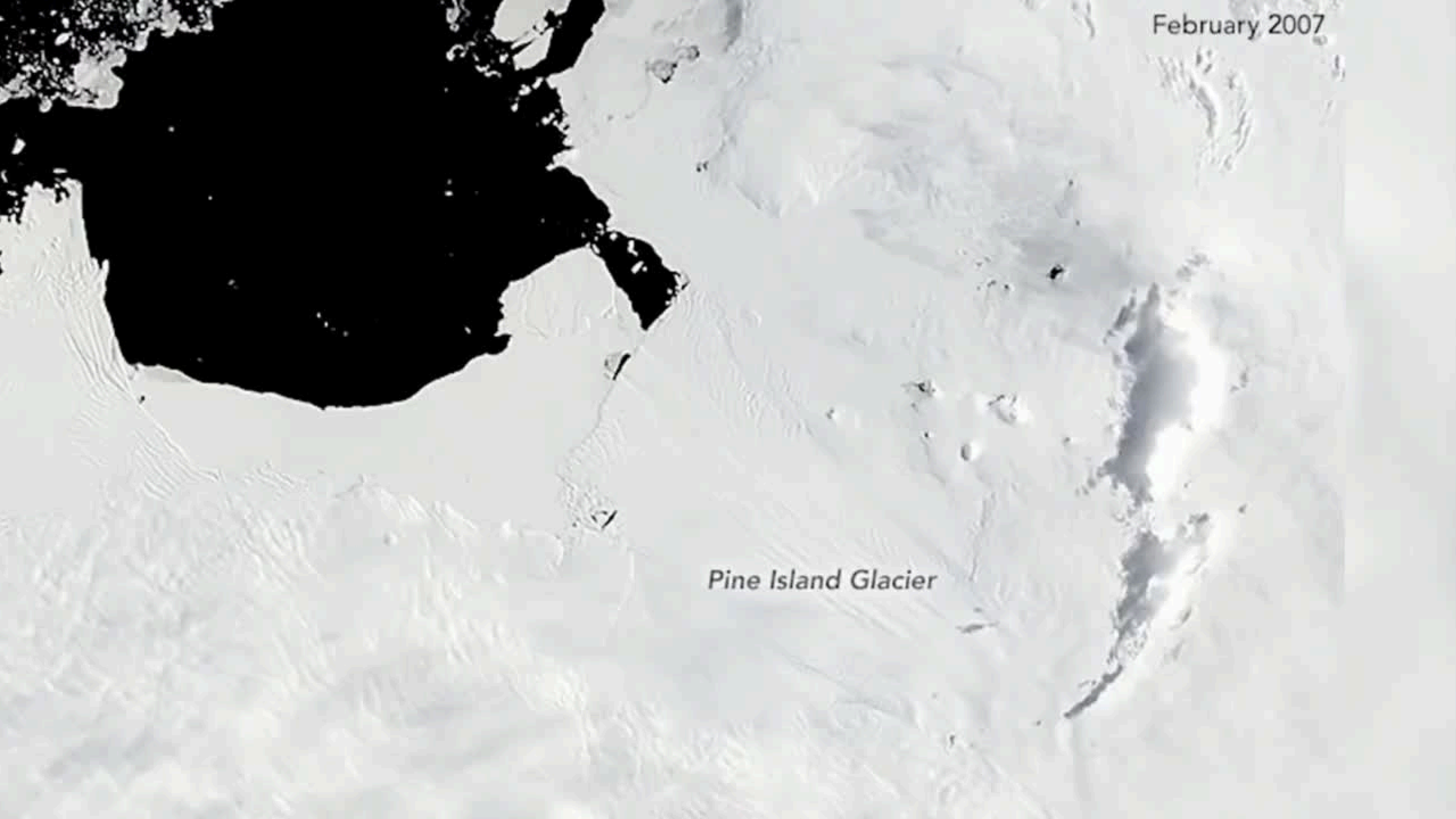


# Vision transformer architecture

1. Split an image into patches
2. Flatten the patches
3. Produce lower-dimensional linear embeddings from the flattened patches
4. Add positional embeddings
5. Feed the sequence as an input to a standard transformer encoder
6. **Pretrain the model with image labels (fully supervised on a huge dataset)**
7. **Finetune on the downstream dataset for image classification**

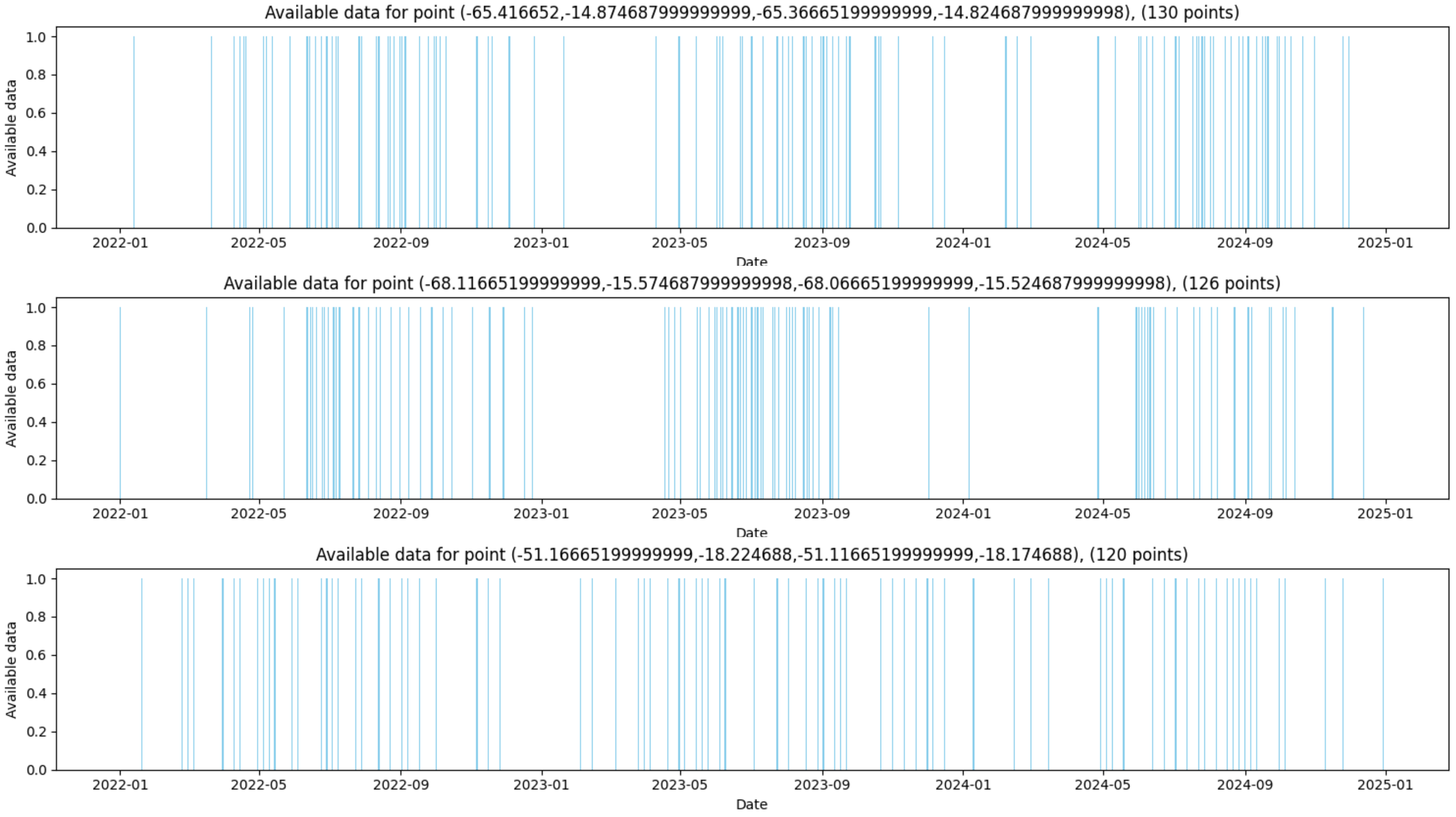


# Satellite Image Time Series


February 2007
Pine Island Glacier

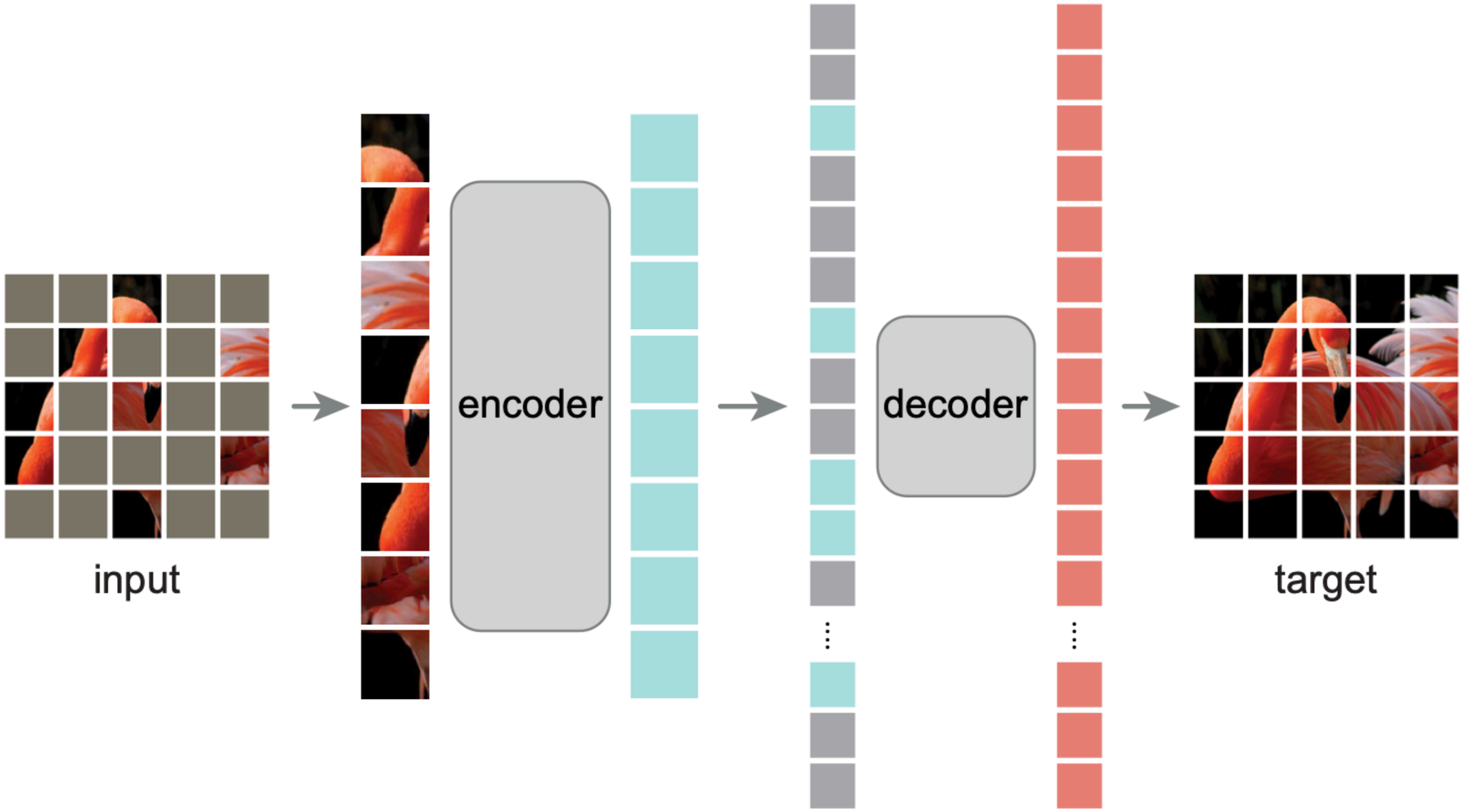
# Advantages of Training Models on SITS

1. Incorporating seasonal variability and changing landscapes over time
2. Applications in Monitoring and Prediction
3. Addressing Data Gaps
4. Environmental Change Detection
5. Reduced sensitivity to noise and variable weather conditions


Available data for point (-65.416652,-14.874687999999999,-65.36665199999999,-14.824687999999998), (130 points)
Available data for point (-68.11665199999999,-15.574687999999998,-68.06665199999999,-15.524687999999998), (126 points)
Available data for point (-51.16665199999999,-18.224688,-51.11665199999999,-18.174688), (120 points)
Available data
Date
0.0
0.2
0.4
0.6
0.8
1.0
2022-01
2022-05
2022-09
2023-01
2023-05
2023-09
2024-01
2024-05
2024-09
2025-01

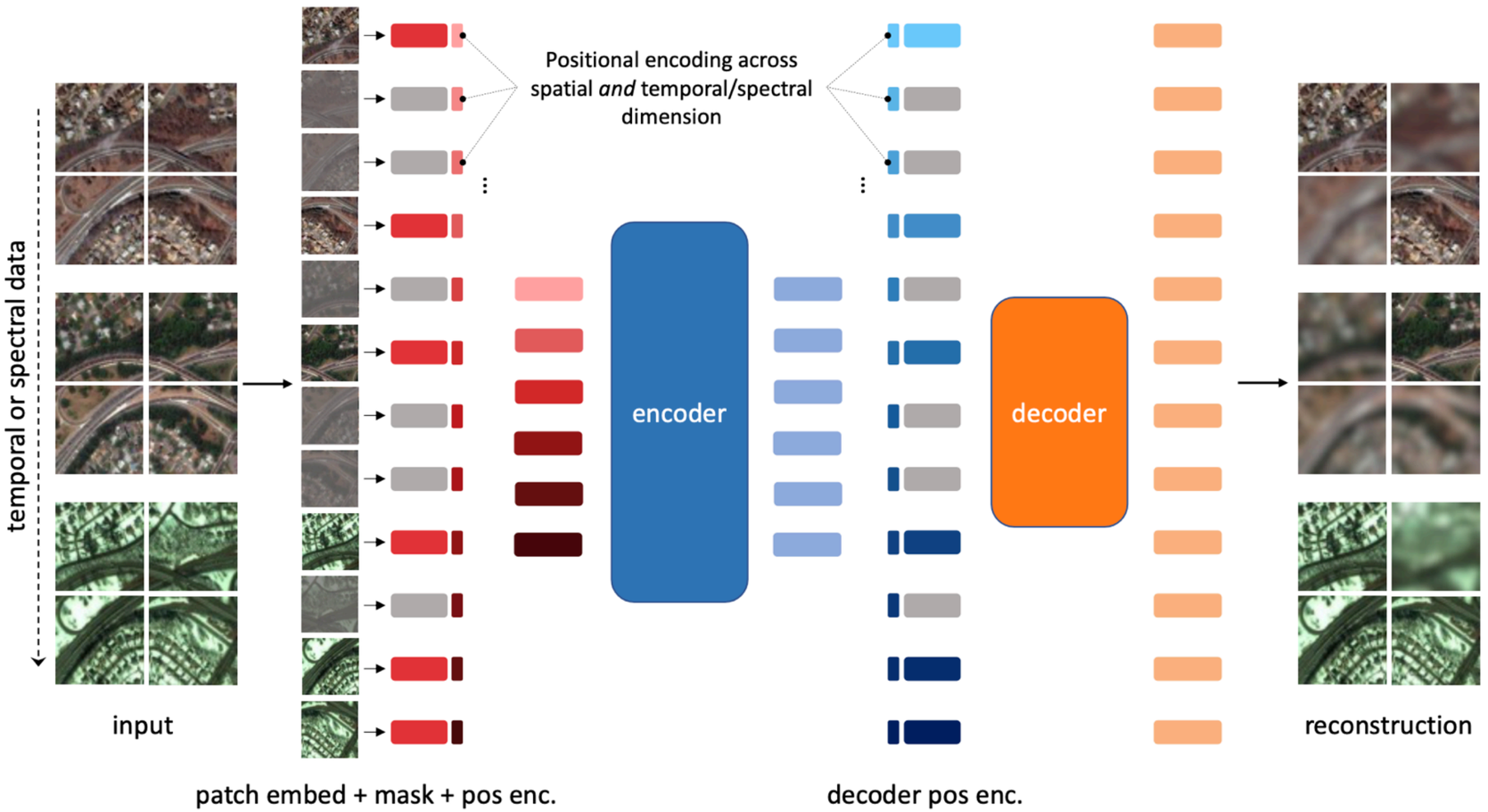
# MAE

# MAE Architecture



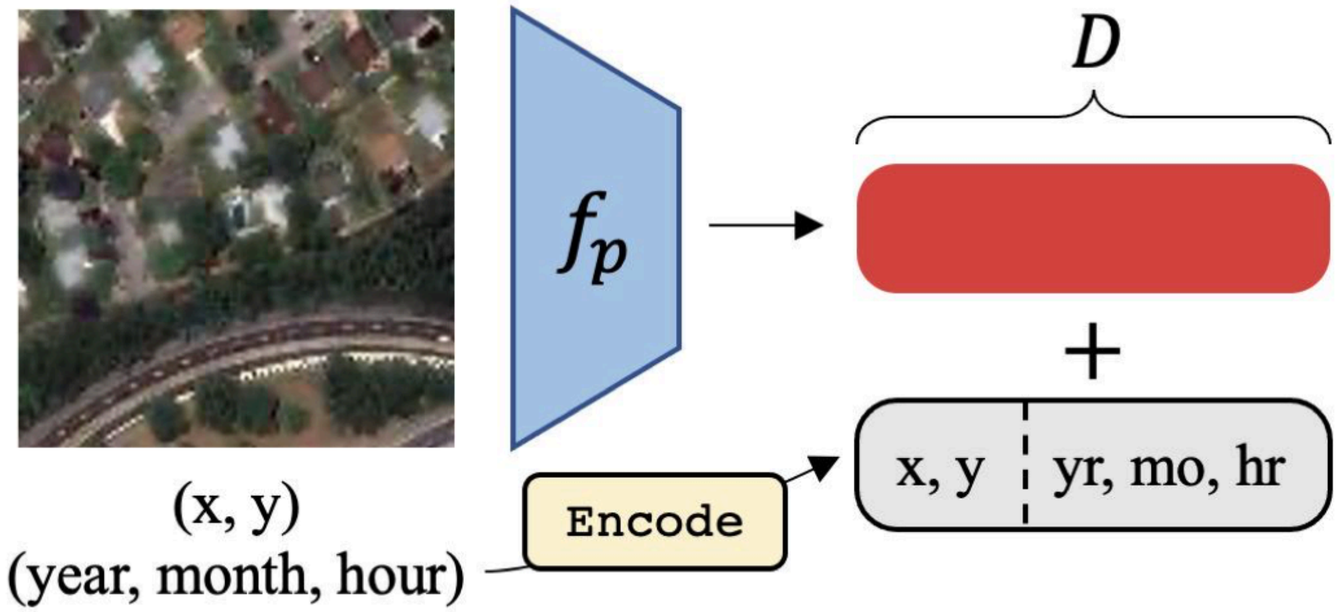
# SatMAE

# SatMAE architecture

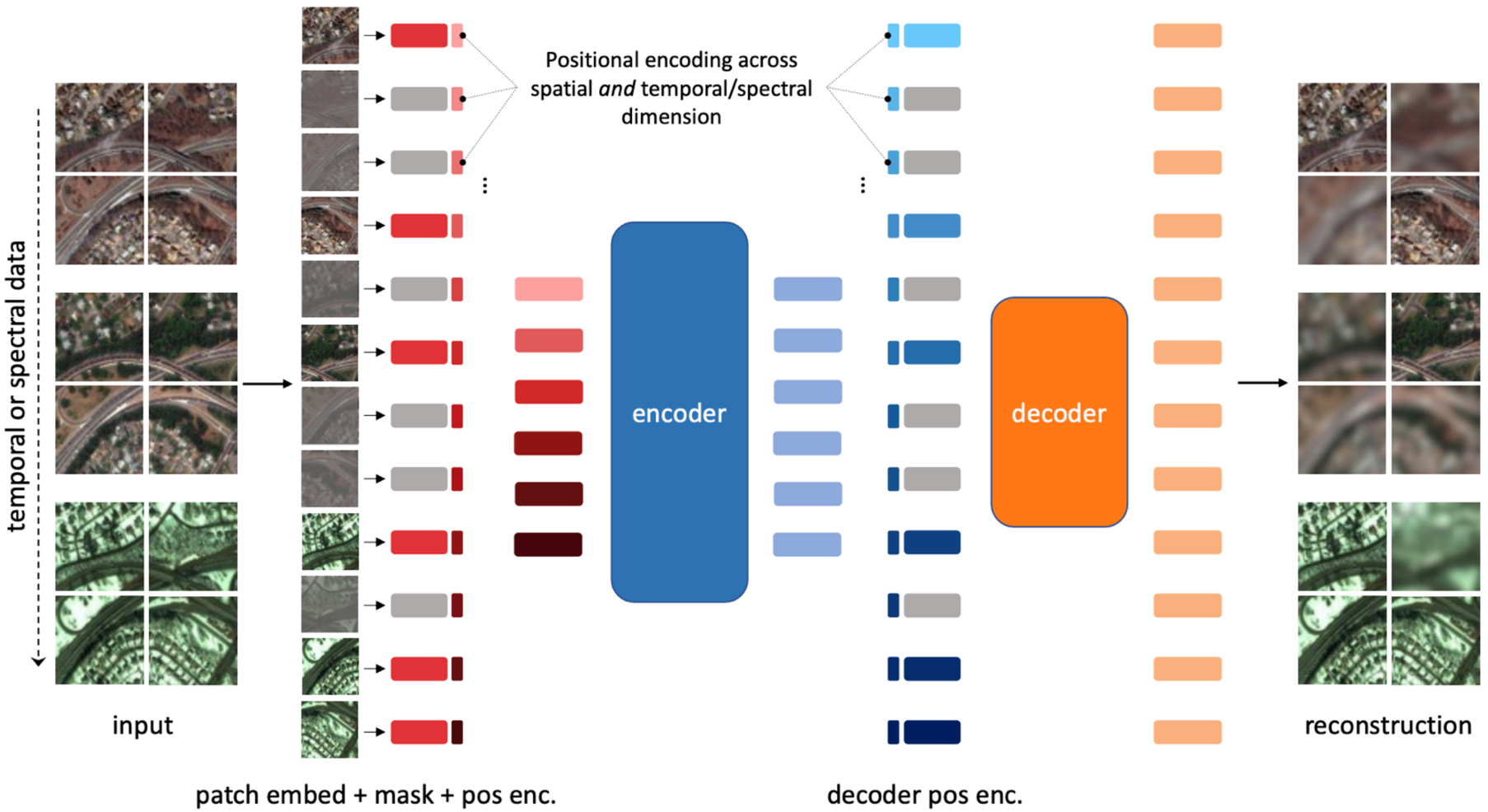


# Temporal Encoding

$$t_{k,i} = CONCAT[Encode(k_{year}, i), Encode(k_{month}, i), Encode(k_{hour}, i)]$$

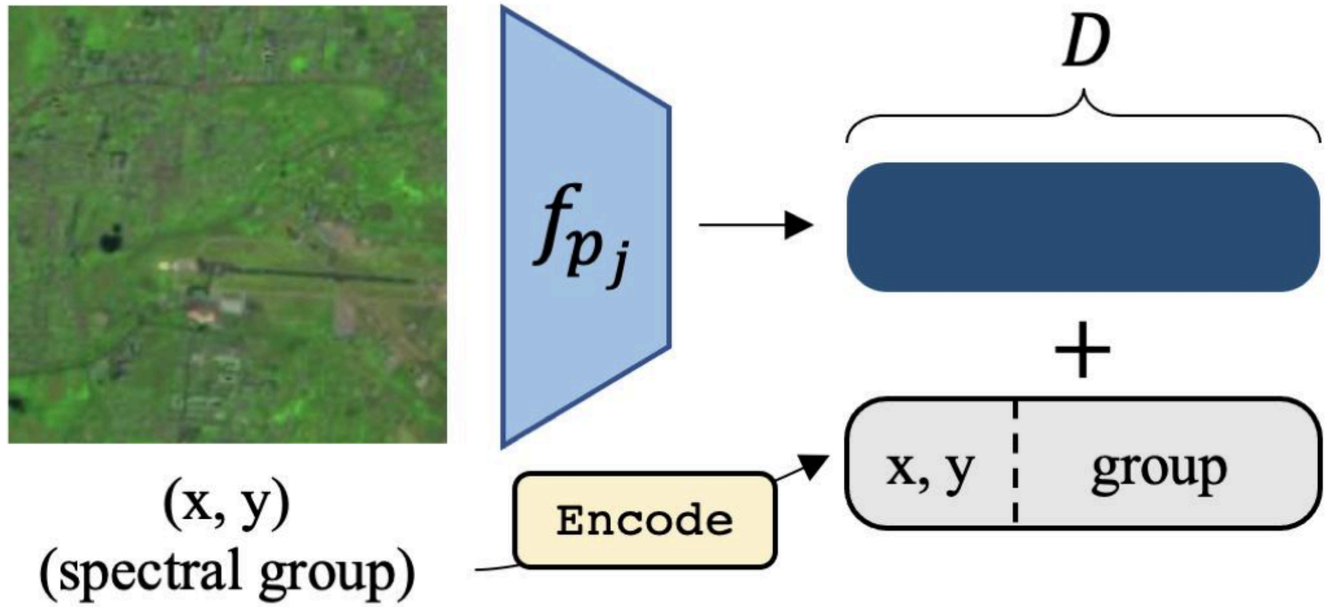


# SatMAE architecture

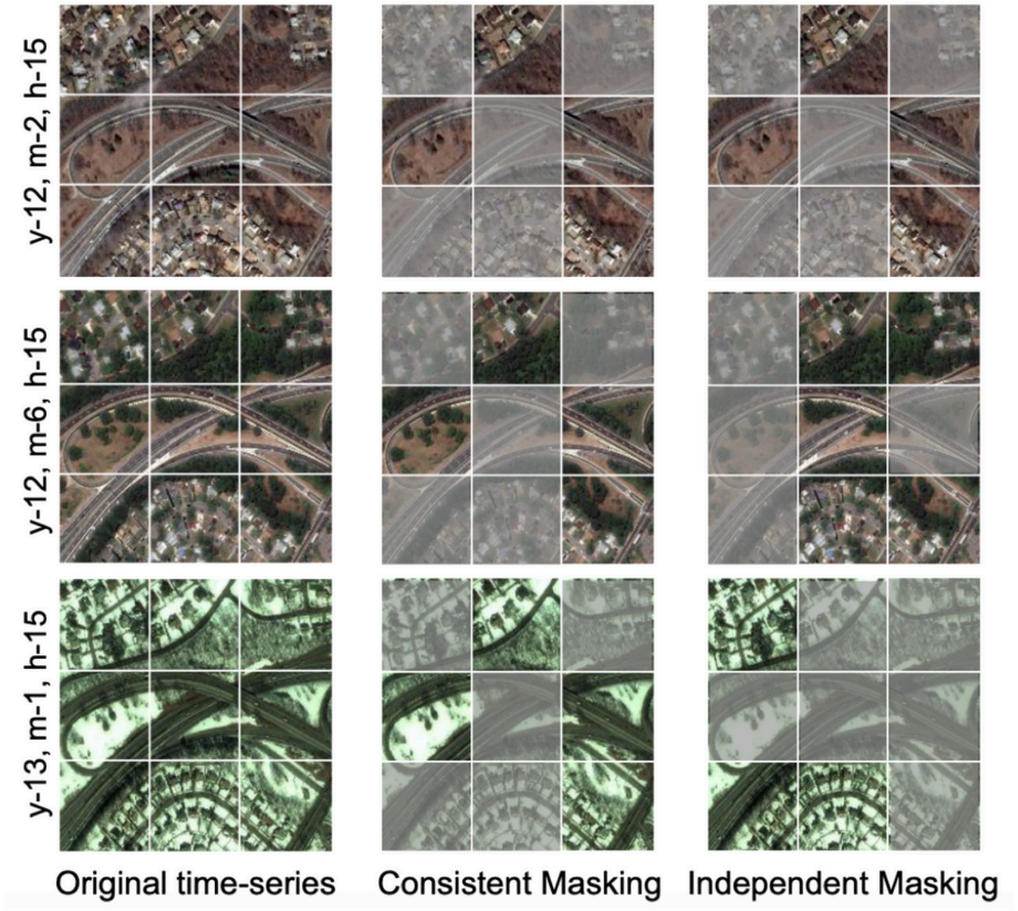


# Spectral Encoding

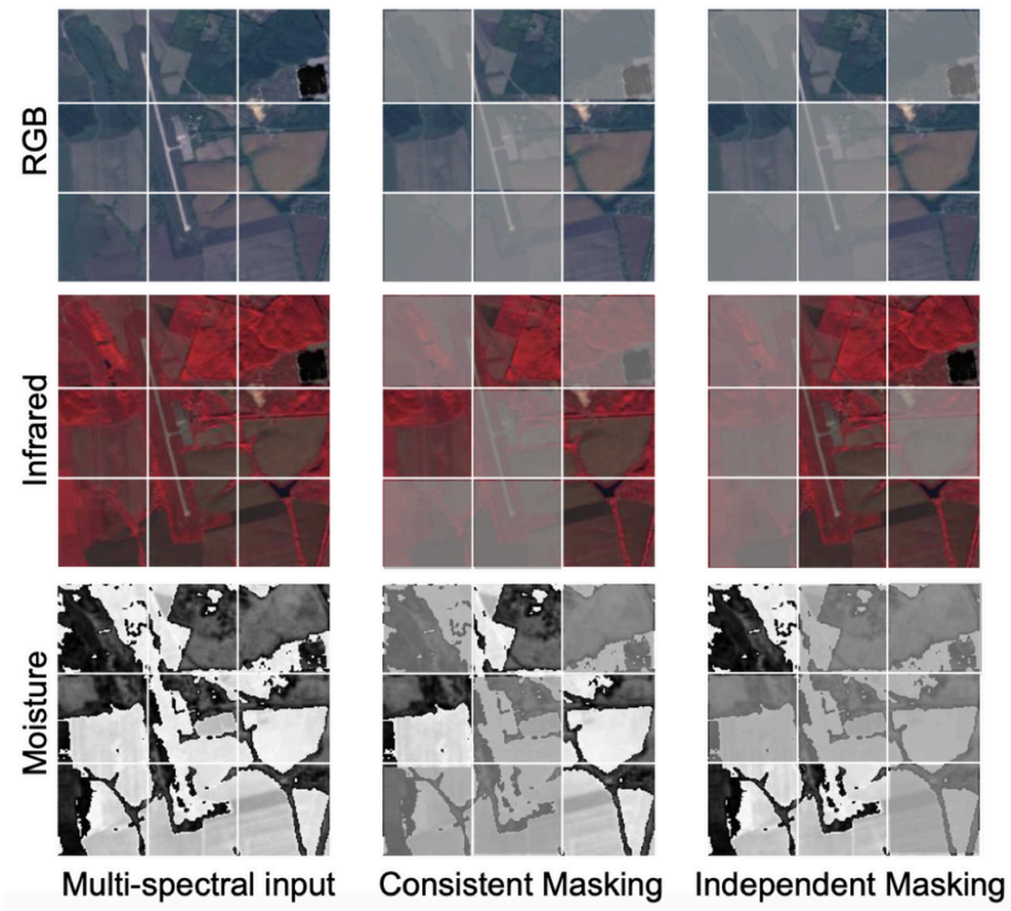
$$g_{k_g,i} = Encode(k_g, i)$$



# Masking

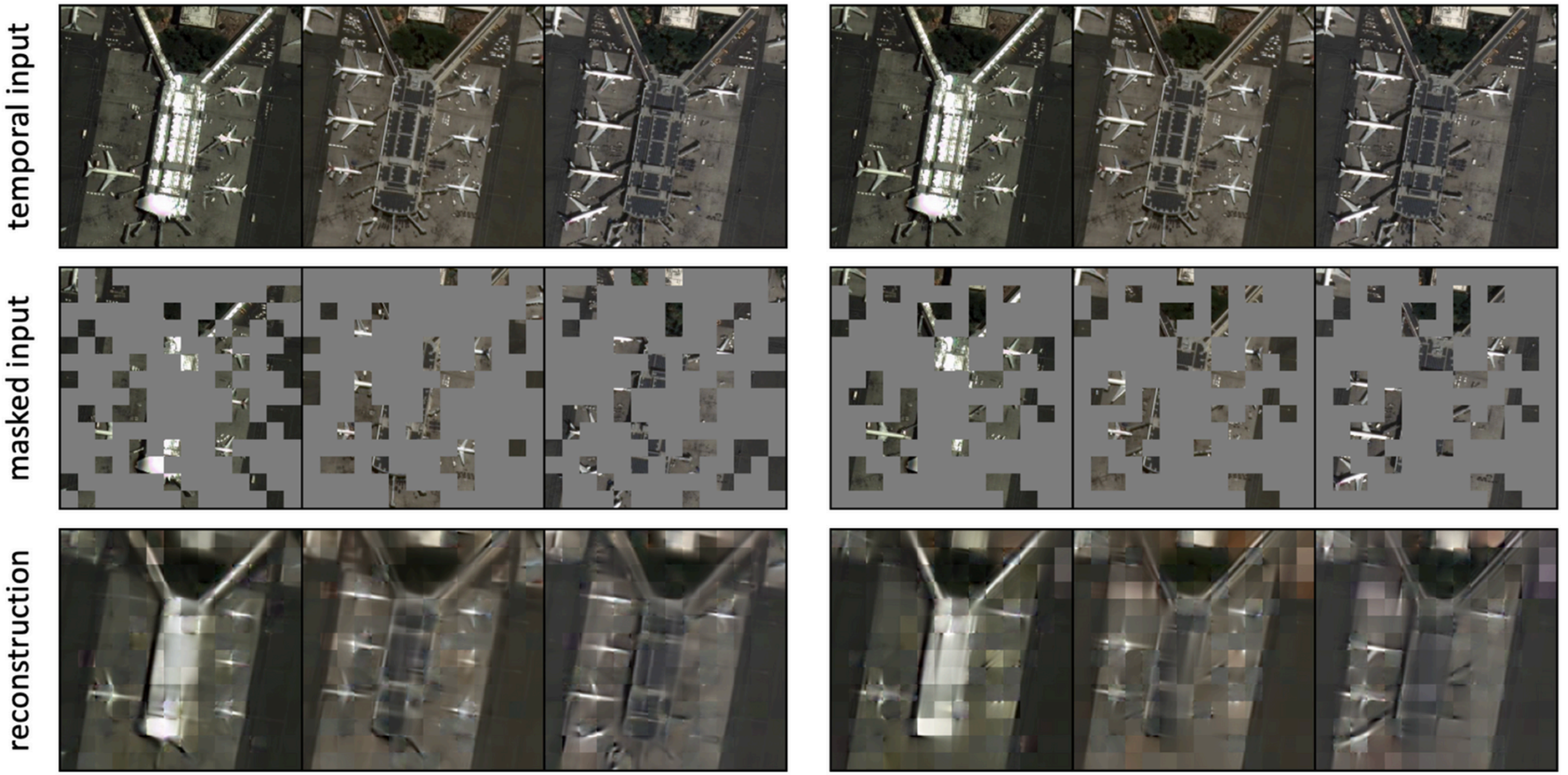



(a) Temporal data



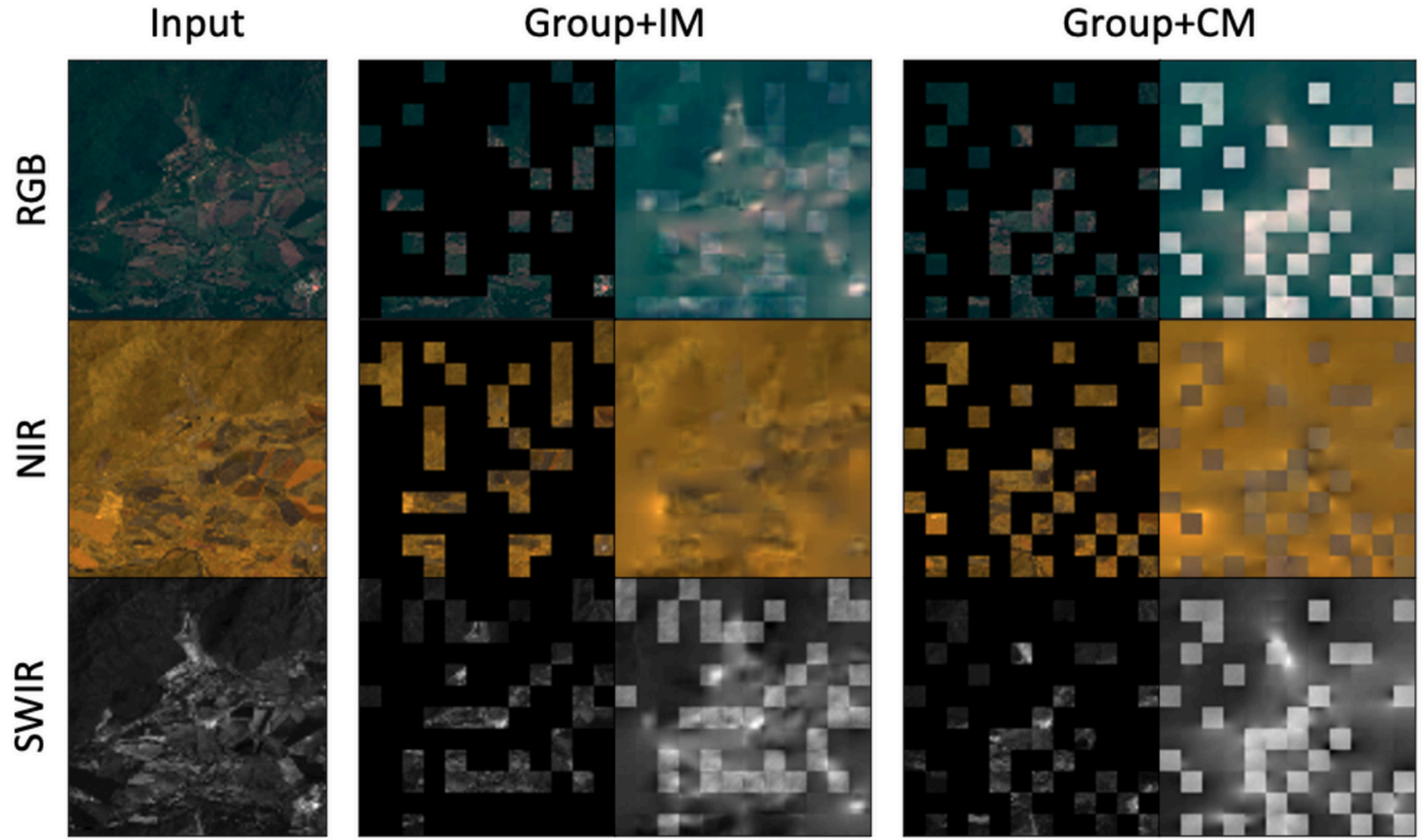

(b) Spectral data

# Visualizing reconstruction quality - temporal SatMAE




Reconstruction quality of SatMAE+IM (left) vs. SatMAE+CM (right)

# Visualizing reconstruction quality - spectral SatMAE




Reconstruction quality of SatMAE+IM (left) vs. SatMAE+CM (right)

# Presto

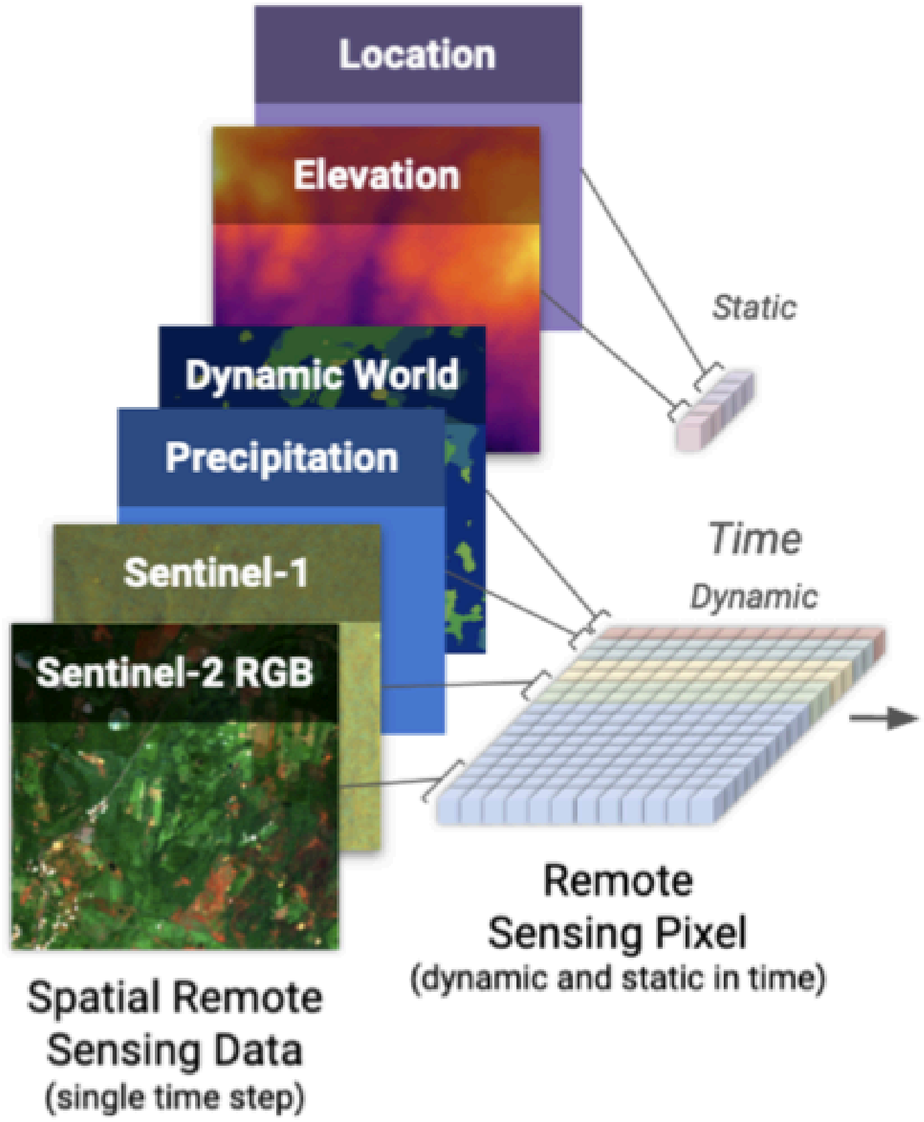
# Presto Input Data

Dynamic variables

- The elevation and slope
- Dynamic World Land Cover classes
- NDVI
- ERA5
- Sentinel-1
- Sentinel-2

Static variables

- Coordinates
- Topography data



# Chanel Groups

- Sentinel-1: The VV and VH bands
- Sentinel-2 RGB: The B2, B3 and B4 bands
- Sentinel-2 Red Edge: The B5, B6 and B7 bands
- Sentinel-2 Near Infra Red (10m): The B8 band
- Sentinel-2 Near Infra Red (20m): The B8A band
- Sentinel-2 Short Wave Infra Red: The B11 and B12
- NDVI: Vegetation index, calculated from the Sentinel-2 B4 and B8 bands.
- ERA5 Climatology: Precipitation and temperature at 2m
- Topography: The elevation and slope of a pixel
- Location: The cartesian coordinates of a pixel

# Variables Encoding

**Topographical data**

$$e^{TG} = h^{TG}(s^{TG}) + [p^{TG}_{channel}; 0; 0]$$

**Coordinates**

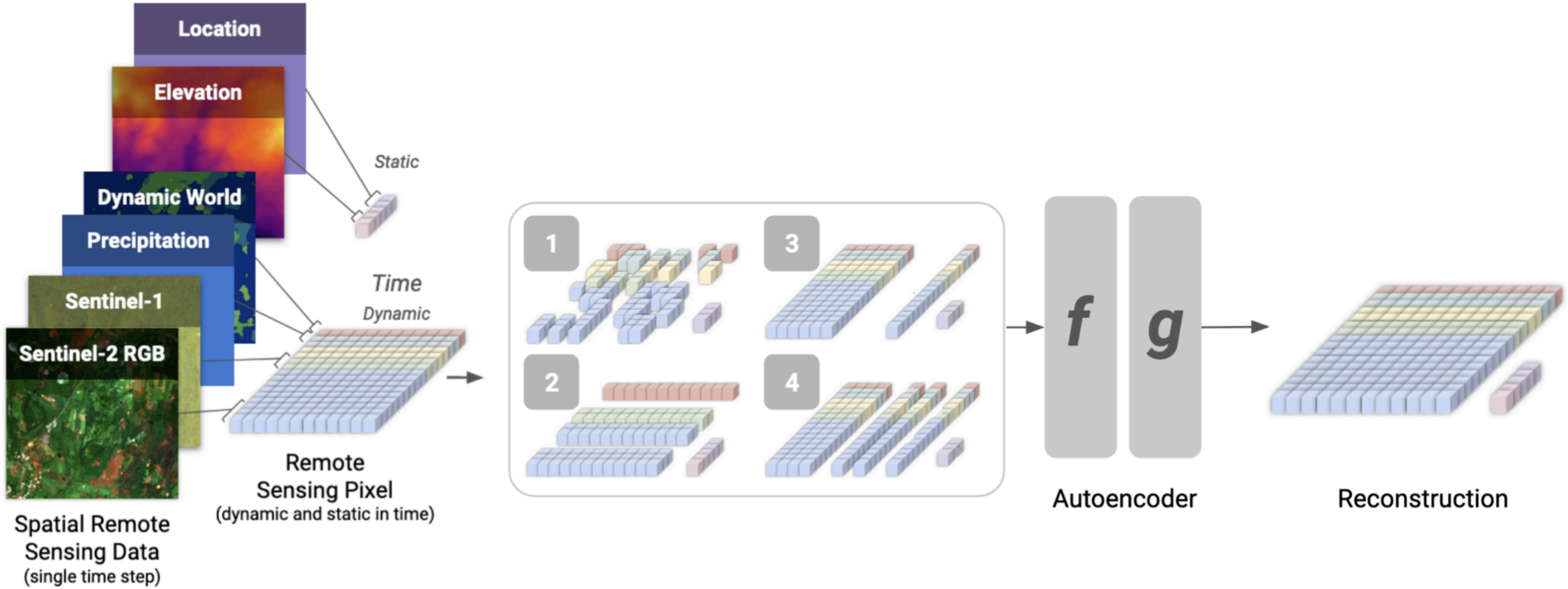
$$e^{Loc} = h^{Loc}(s^{Loc})$$

**Dynamic variables**

$$e^c_i = h^c(t^c_i) + [p^c_{channel}; p_{sin(i)}; p_{month(i)}]$$

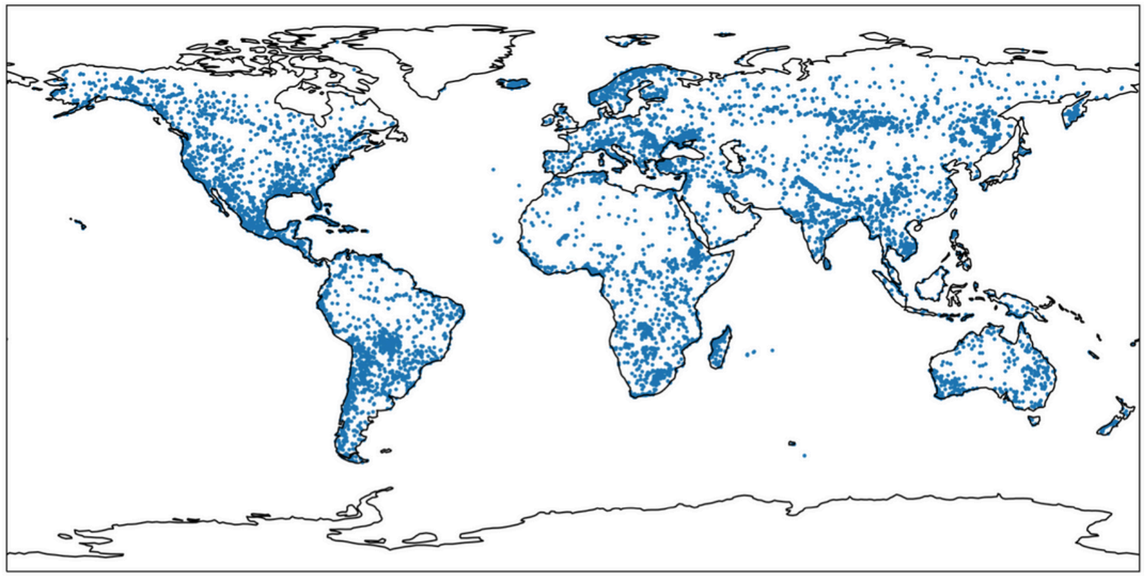
where:
$h$ : linear projection
*sin*: sinusoidal positional encoding
*channel*: nn.Embedding
*month*: $p(month, 2i) = sin((2\pi \times month)/12)$
$p(month, 2i+1) = cos((2\pi \times month)/12)$

# Presto architecture



# Pre-training Dataset

- **Geographic Distribution:** Earth divided into three regions – Western Hemisphere and two Eastern Hemisphere regions, further subdivided into ecoregions.
- **Sampling Strategy:** Stratified sampling based on land cover classes.
- **Resolution:** Each sample is a 510 × 510 pixel tile with 10-meter spatial resolution.
- **Pixel Time-series:** 2,500 pixels sampled per tile, resulting in 21,535,000 pixel samples, each with 24 monthly time steps.



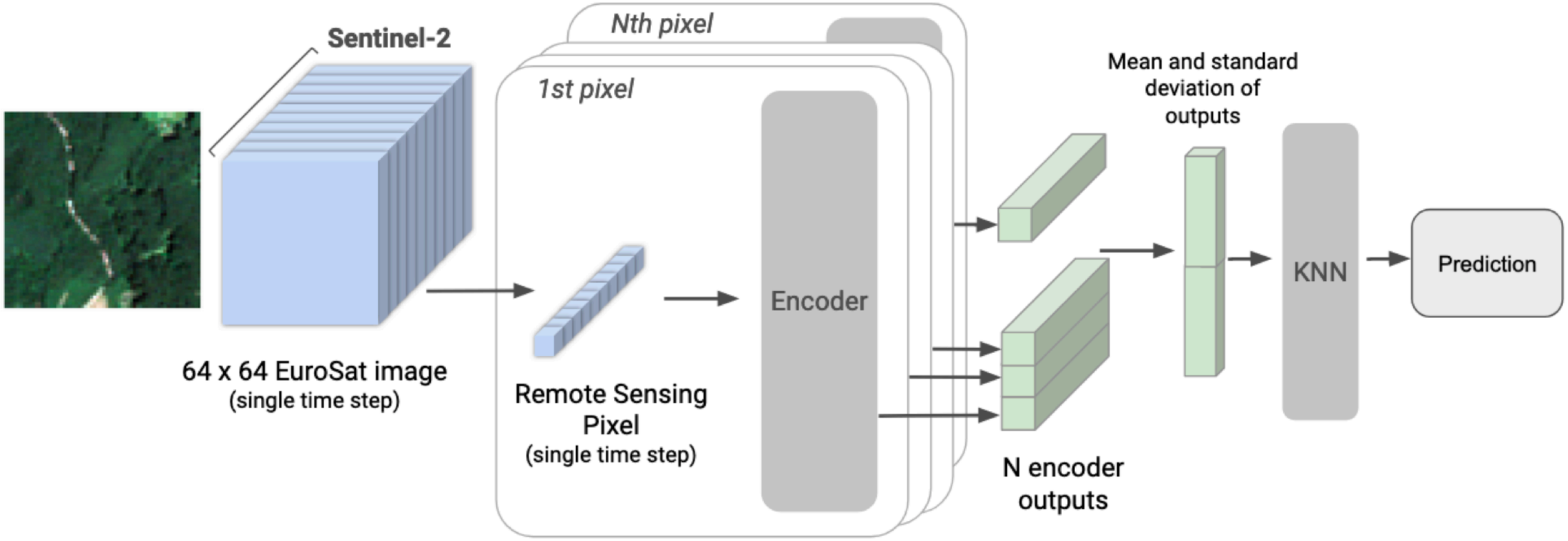
The distribution of the pre-training dataset

# Training

$$L_{total} = L_{MSE} + \lambda \frac{N_{cat}}{N_{cont}} L_{CE}$$

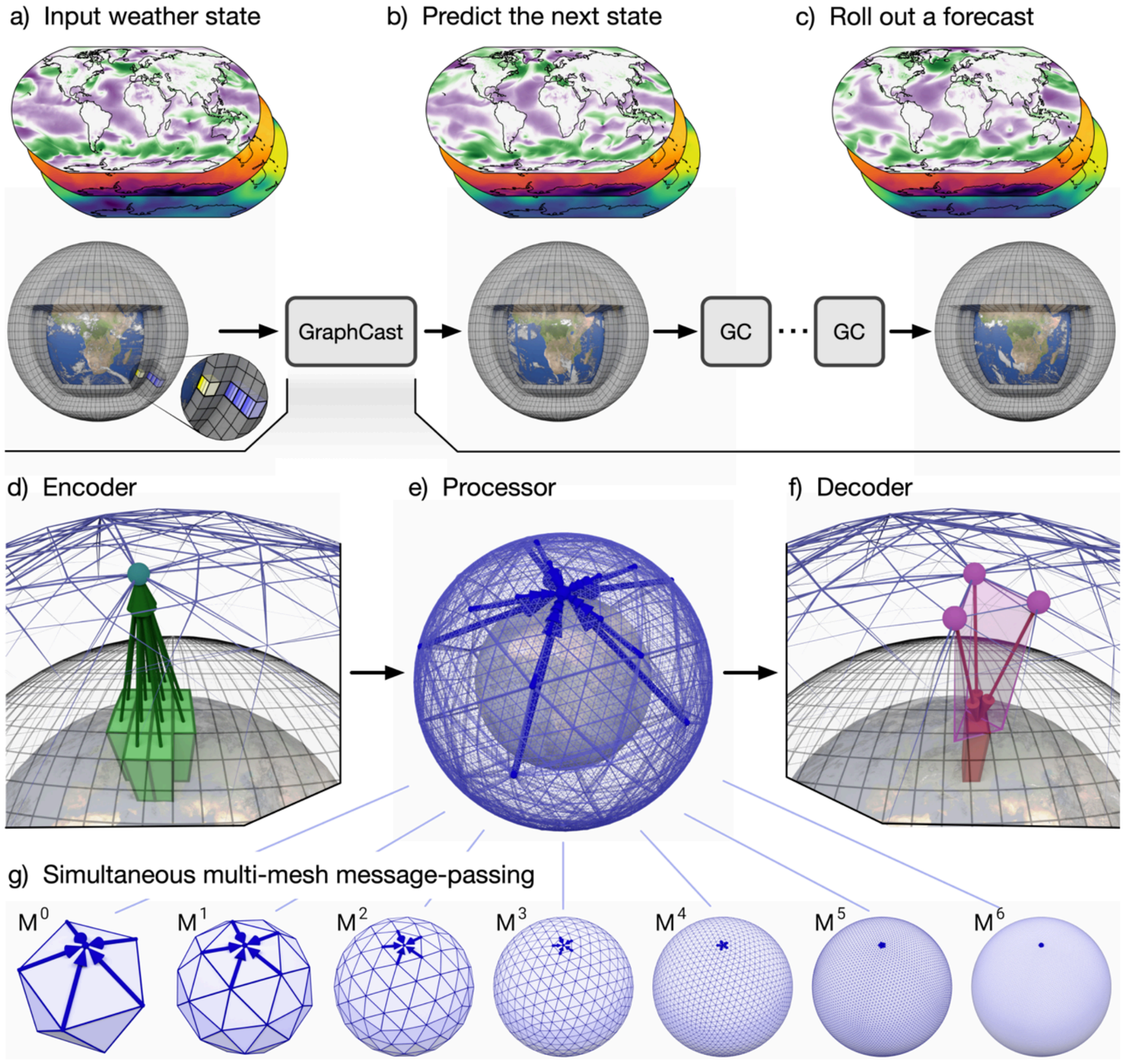
L_{MSE} mean squared error reconstruction loss used for the continuous values (f.eg month)
L_{CE} cross entropy loss used for the categorical values (f.eg Topographical data)
N_{count} number of masked continuous values
N_{cat} number of masked categorical values in the batch
λ hyperparameter

# How to use Presto encoder?




Notebook: downstream_task_demo.ipynb

# GraphCast


a) Input weather state
b) Predict the next state
c) Roll out a forecast
GraphCast
GC
GC
d) Encoder
e) Processor
f) Decoder
g) Simultaneous multi-mesh message-passing
M0
M1
M2
M3
M4
M5
M6

# Case Studies

## Case Studies – Google Earth Engine

Google Earth Engine combines a multi-petabyte catalog of satellite imagery and geospatial dataset...

GoogleEarth

U-Net: Convolutional Networks for Biomedical Image Segmentation

AN IMAGE IS WORTH 16X16 WORDS:
TRANSFORMERS FOR IMAGE RECOGNITION AT SCALE

Lightweight, Pre-trained Transformers for Remote Sensing Timeseries

SatMAE: Pre-training Transformers for Temporal and Multi-Spectral Satellite Imagery

Masked Autoencoders Are Scalable Vision Learners

GraphCast: Learning skillful medium-rangeglobal weather forecasting

Laboratory of Geo-information Science and Remote Sensing

ESA

NASA