LECTURE 3

MACHINE LEARNING & DEEP LEARNING FOUNDATIONS

Geospatial Representation Learning

PRESS → OR SPACE.

UNIVERSITÄT BONN

Today In One Sentence

Deep learning learns useful tensor transformations from data.

Inputs

geospatial tensors

Models

architectures with parameters

Learning

optimization from examples

Why This Lecture Matters

Later course topics reuse the same foundation:

- self-supervised learning
- multimodal foundation models
- Earth embeddings
- location encoders
- implicit neural representations
- geospatial AI agents

The vocabulary begins with tensors, architectures, losses, gradients, and generalization.

Test Figure: Manual Assembly

This slide is a sandbox for reusable draggable figure elements.

x

z

Input Image

`[H × W × 13]`

Sentinel-2 patch

z

learned representation

Encoder

fθ

Conv ReLU Pool

v-drag test

MODEL

`z = fθ(x)`

plain text formula box

embedding

encode

Lecture Structure

1	2	3	4	5
Opening and geodata recap	Evolution of deep learning	Deep model architectures	Learning algorithms	Wrapup and course bridge

Recap: Geodata Is Structured

From Lecture 2:

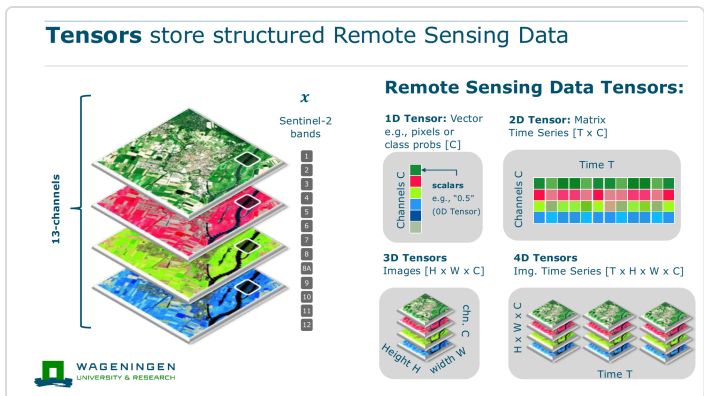
- spatial reference
- spatial resolution
- temporal resolution
- spectral resolution
- uncertainty and missingness
- sampling bias

Deep learning does not remove these properties.

It consumes them through tensor representations and inherits their biases.

Tensors: The Common Interface

A tensor is a multi-dimensional array.



Common tensor forms:

- scalar: one value
- vector: one spectrum or feature vector
- matrix: one raster band
- image tensor: `[H x W x C]`
- time series: `[T x C]`
- image time series: `[T x H x W x C]`

Pixel Spectrum

One multispectral pixel is a vector.

$$x_{pixel} = \begin{bmatrix} B02 & B03 & B04 & B08 & B11 & B12 \end{bmatrix}$$

The values are numbers for the model, but we interpret them as physical measurements related to vegetation, water, soil, snow, or built surfaces.

Image Tensor

A multispectral image patch is often:

$$x \in \mathbb{R}^{H \times W \times C}$$

H	W	C
rows	columns	bands or channels

Time And Space Together

Repeated observations add a temporal axis.

$$x \in \mathbb{R}^{T \times H \times W \times C}$$

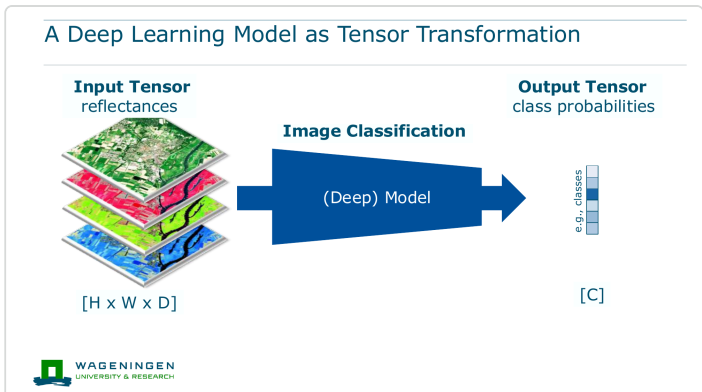
This is common in crop mapping, forest monitoring, flood mapping, and land-cover change detection.

Learning Means Tensor Transformation

A deep model maps an input tensor to an output tensor.

$$\hat{y} = f_W(x)$$

- `x` : input tensor
- `f_W` : model with learned parameters
- `ŷ` : prediction tensor



Opening Takeaway

Deep learning for geospatial data begins with a simple abstraction:

> **geospatial measurements become tensors, and models learn transformations between tensors.**

Everything else today explains what those transformations look like and how they are learned.

Why Deep Learning Emerged

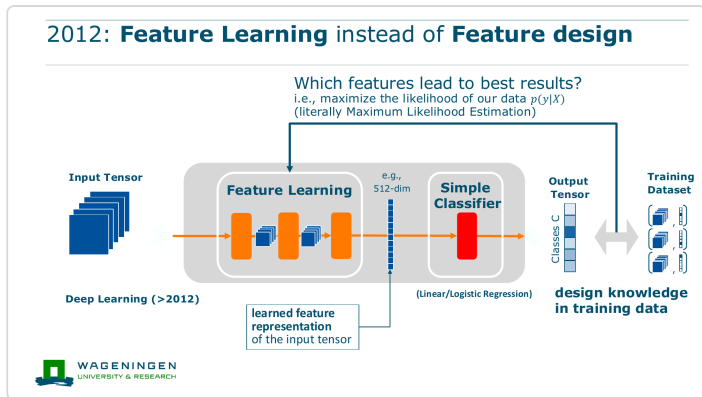
Classic machine learning and deep learning differ in where design knowledge is placed.

Classic ML:

```
features → classifier
```

Deep learning:

```
representation learning → prediction
```



Before 2012: Classic ML

Remote sensing used many hand-designed features.

Spectral indices

NDVI, NDWI, NDBI

Texture features

local spatial structure

Classifiers

SVMs, random forests, logistic regression

Strength

effective with small labelled datasets

Classic ML Limitation

Feature design is powerful, but manual.

If the relevant pattern is complex, the modeller must already know how to express it as a feature.

Examples: crop phenology, urban morphology, flood context, forest structure, multimodal sensor interactions.

2015 Onwards: Supervised Deep Learning

Deep networks learn features from labelled data.

More data

large labelled benchmarks

More compute

GPU training

Better architectures

CNNs, ResNets, Transformers

Supervised Deep Learning Workflow

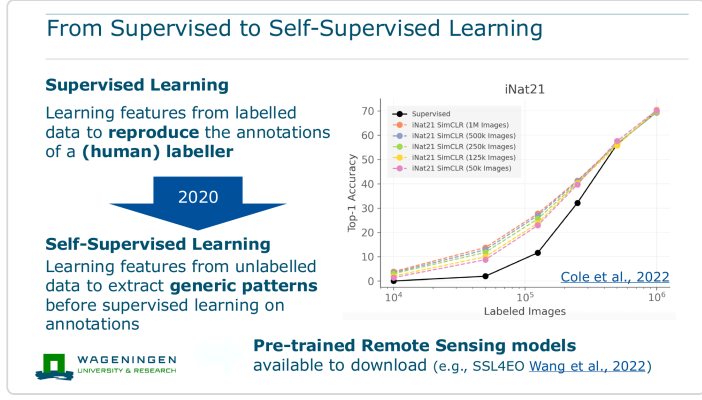
The labelled dataset tells the model what predictions should look like.

From Labels To Pretraining

Labelled data is expensive in geospatial domains.

Manual labels may require:

- expert interpretation
- field campaigns
- harmonized taxonomies
- temporal alignment
- quality control



Self-Supervised Learning

Self-supervised learning learns from structure in unlabelled data.

Supervised

match human-provided labels

Self-supervised

solve a pretext task from the data itself

This is the bridge to foundation models in Lecture 4.

Evolution Takeaway

The evolution is a shift in what we design.

Period	Main design effort	Typical data
Classic ML	features	small labelled datasets
Supervised DL	architectures	large labelled datasets
Self-supervised DL	objectives	large unlabelled or multimodal archives

Part 1: Deep Model Architectures

Architecture answers:

How should a model transform tensors?

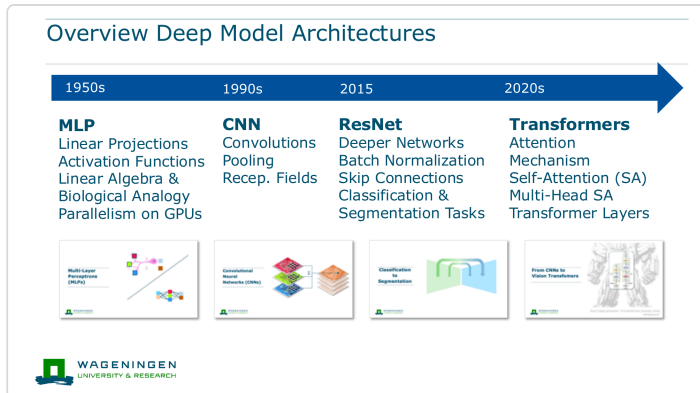
building blocks	network families	inductive biases

Architecture Families

Today we focus on:

- MLPs
- CNNs
- Transformers

Each family makes different assumptions about the input tensor.



Parameters

Parameters are the numbers the model learns.

Before training:

`W` is mostly random.

After training:

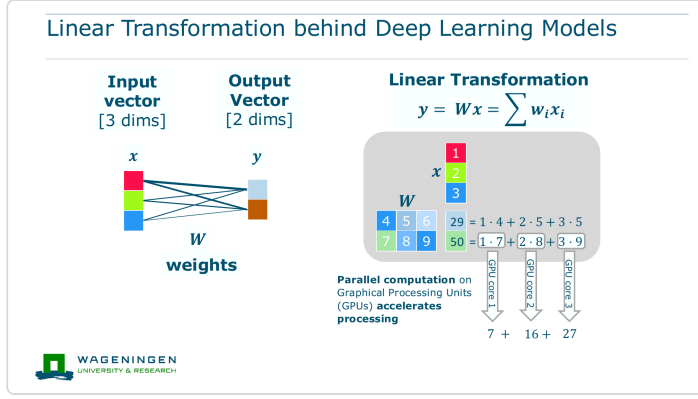
`W` contains useful transformations.

Linear Layer

The basic operation is often:

$$h = Wx + b$$

- `x` : input vector
- `W` : weights
- `b` : bias
- `h` : output vector



Bias Term

The bias shifts the output.

$$h = Wx + b$$

Without `b`, the transformation must pass through the origin. With `b`, the model has more flexibility.

Activation Function

Activation functions add nonlinearity.

Without activations

stacked linear layers collapse into one linear map

With activations

networks model complex functions

Pooling

Pooling reduces spatial or temporal resolution.

Common idea:

```
local neighborhood → summary value
```

Pooling can make features less sensitive to small shifts, but it also discards detail.

Dropout

Dropout randomly removes activations during training.

Goal

reduce dependence on any single feature

Use

regularization in neural networks

Normalization

Normalization stabilizes training by controlling feature scales.

Examples:

- batch normalization
- layer normalization

Normalization is one reason very deep networks became easier to train.

Residual Connections

Residual connections let layers learn changes rather than full transformations.

$$h_{next} = h + F(h)$$

They help gradients flow through deep networks and are central in ResNets and Transformers.

Multi-Layer Perceptron

An MLP stacks dense layers.

MLPs are useful for vectors: pixel spectra, tabular covariates, coordinates, and embeddings.

MLP Limitation

An MLP treats its input as a vector.

If we flatten an image, the model no longer directly knows which pixels are neighbors.

This motivates architectures with spatial structure.

Convolutional Neural Network

A CNN applies local filters across an image.

Key ideas:

- local receptive fields
- shared weights
- feature maps

CNNs encode the assumption that nearby pixels are related and patterns can repeat across space.

CNNs In Remote Sensing

CNNs learn spatial patterns such as:

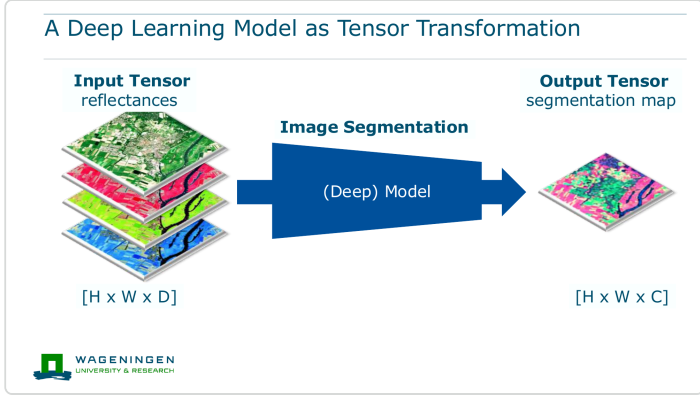
- field boundaries
- building shapes
- road networks
- urban and forest texture

Segmentation With CNNs

Semantic segmentation predicts a label for every pixel.

$$f_W(x) : \mathbb{R}^{H \times W \times C} \rightarrow \mathbb{R}^{H \times W \times K}$$

Example: land cover, buildings, roads, water, burned area.



Transformer Intuition

Transformers process tokens.

patch	patch	date	sensor	text

Tokens can represent image patches, dates, modalities, coordinates, or words.

Self-Attention

Self-attention asks:

> **Which other tokens are relevant for this token?**

In geospatial data, attention can connect distant image patches, different dates, or different sensor modalities.

Transformers In Geospatial AI

Transformers are useful when models must combine:

- image patches
- time series
- optical, SAR, climate, metadata
- text and task descriptions

Architecture Comparison

Architecture	Input assumption	Geospatial fit
MLP	vector features	spectra, tabular covariates, embeddings
CNN	local spatial structure	image classification and segmentation
Transformer	token interactions	temporal, multimodal, foundation models

Architecture Takeaway

Architectures are not only implementation details.

They encode assumptions about what structure matters:

vectors	local neighborhoods	global token relations

Part 2: Learning Algorithms

Now we ask:

> **How do the parameters of a deep architecture become useful?**

The answer is optimization from data.

Supervised Dataset

We start with labelled examples:

$$\mathcal{D} = \{(x_i, y_i)\}_{i=1}^{N}$$

x_i

input tensor

y_i

target or label

Prediction And Loss

The model predicts:

$$\hat{y}_i = f_W(x_i)$$

The loss measures mismatch:

$$\mathcal{L}(\hat{y}_i, y_i)$$

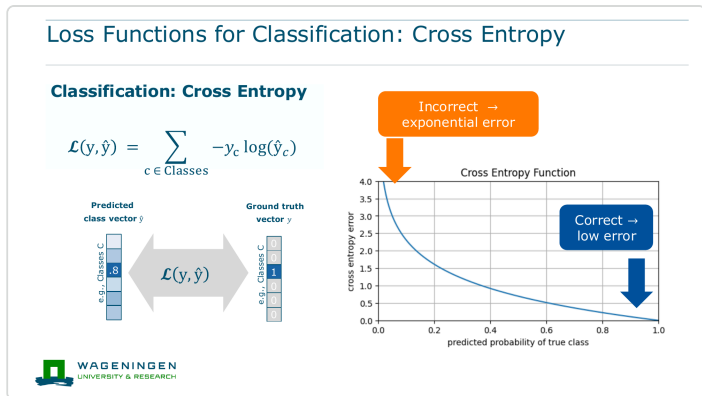
Training uses this scalar loss to decide how to update the parameters.

Cross Entropy

Cross entropy is common for classification.

It penalizes confident wrong predictions strongly.

Example: predicting `urban: 0.95` when the target is `water`.

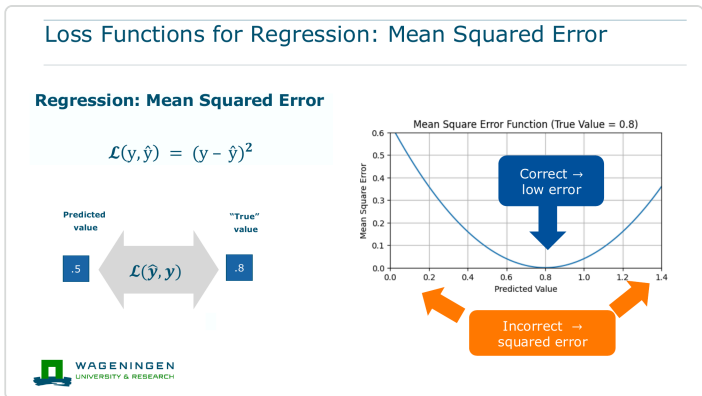


Mean Squared Error

Mean squared error is common for regression.

$$\mathcal{L}(\hat{y}, y) = (\hat{y} - y)^2$$

Example: biomass, soil carbon, temperature, flood depth.



Training Objective

Training searches for parameters with low loss.

$$W^* = \arg\min_W \mathcal{L}(f_W(x), y)$$

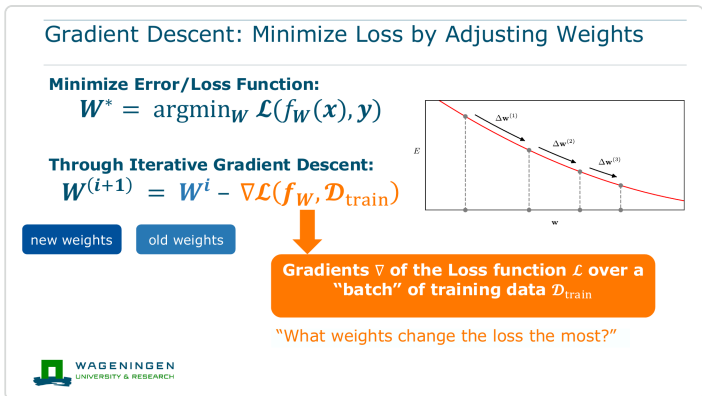
Read this as: find parameters that make predictions match examples.

Gradient Descent

Gradient descent updates parameters by moving downhill.

$$W_{t+1} = W_t - \eta \nabla_W \mathcal{L}$$

- η : learning rate
- ∇L : direction of steepest increase
- negative sign: move toward lower loss



Loss Surface

The loss changes as parameters change.

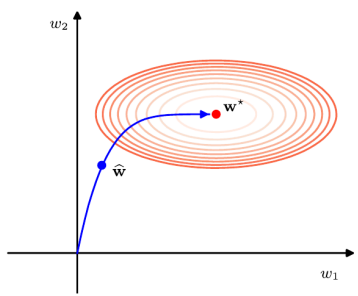
Minimizing error in a high-dimensional Loss Surface

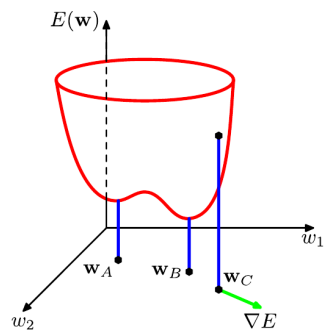
Minimize Error/Loss Function:

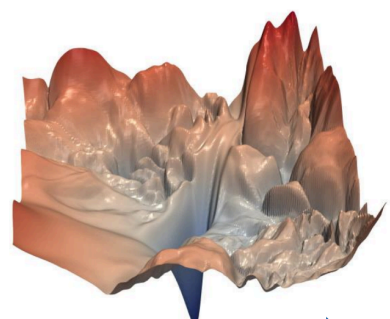
$$W^* = \text{argmin}_W \mathcal{L}(f_W(x), y)$$

Through Iterative Gradient Descent:

$$W^{(i+1)} = W^i - \nabla\mathcal{L}(f_W, \mathcal{D}_{\text{train}})$$







WAGENINGEN UNIVERSITY & RESEARCH

more complex models have more complex loss surfaces

Learning Rate

The learning rate controls step size.

Too small

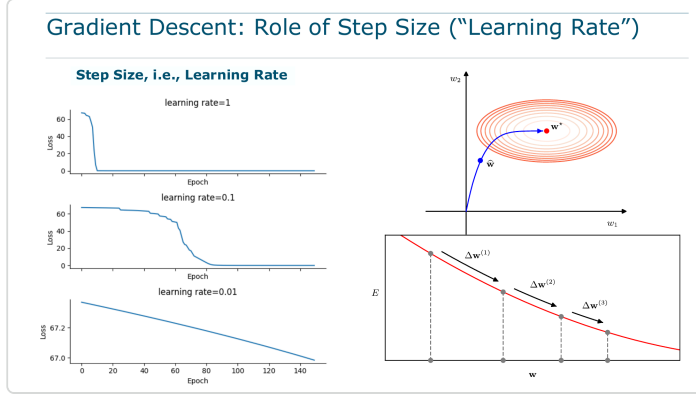
slow learning

Useful range

steady improvement

Too large

unstable training

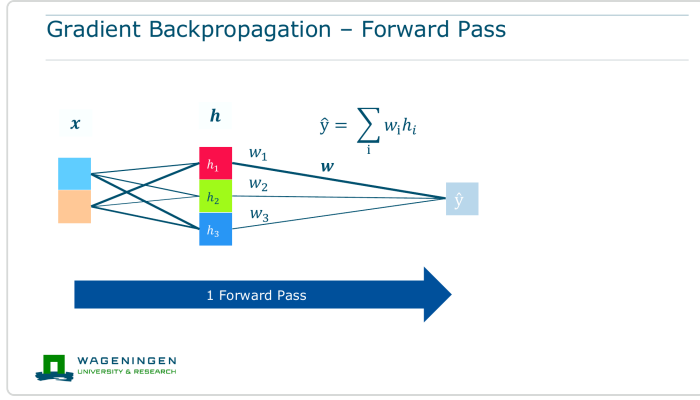


Backpropagation

Backpropagation computes gradients through many layers.

It applies the chain rule from the loss backward through the computation graph.

This tells each parameter how it influenced the final error.



Minimal PyTorch Loop

```
y_pred = model(x)
loss = loss_fn(y_pred, y)

loss.backward()
optimizer.step()
optimizer.zero_grad()
```

This is the operational core of supervised deep learning.

Train, Validation, Test

Training

updates parameters

Validation

chooses settings and stopping point

Test

estimates final performance

Generalization

Generalization means performance on new data.

Training performance:

how well the model fits seen examples.

Test performance:

how well the model works on unseen examples.

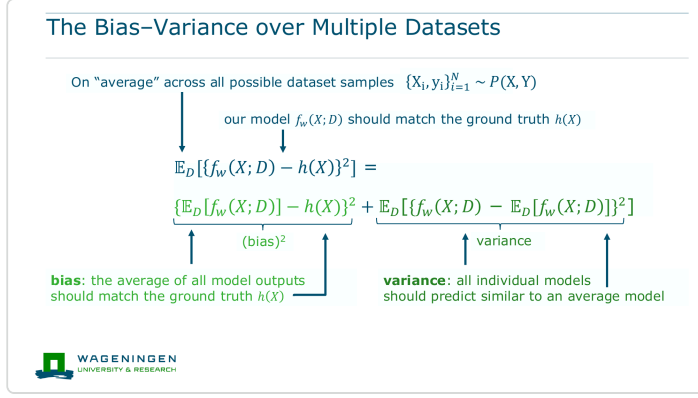
Bias And Variance

High bias

model too simple

High variance

model too sensitive to samples



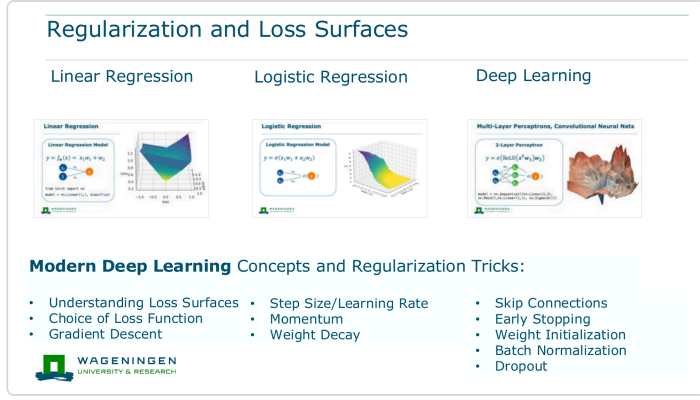
Deep models often have enough capacity to overfit.

Regularization

Regularization discourages brittle solutions.

Examples:

- weight decay
- dropout
- data augmentation
- early stopping
- normalization



Geospatial Validation Is Special

Random splits can overestimate performance.

Nearby samples often look similar because of spatial autocorrelation.

Random splits can place nearly duplicate landscapes in train and test.

Spatial, temporal, or region-based splits often provide a more honest estimate of generalization.

Distribution Shift

Distribution shift means training and deployment data differ.

$$p_{train}(x, y) \neq p_{test}(x, y)$$

- new region
- new season or year
- new sensor or resolution
- new climate or land-use regime

Geospatial Shift Example

A crop model trained in one region may learn a shortcut.

Training region:

irrigated cropland has a strong summer green signal.

New region:

rainfed cropland has different phenology and soils.

Learning Takeaway

Learning algorithms connect data, architecture, and objective.

data → loss → gradients

The real test is not fitting the training set, but generalizing under geospatial shift.

What To Remember

- Geodata enters deep learning as tensors.
- Deep learning learns tensor transformations.
- Architectures encode assumptions about structure.
- MLPs, CNNs, and Transformers are different transformation families.
- Losses define what "better" means.
- Gradient descent updates parameters.
- Backpropagation computes gradients efficiently.
- Geospatial generalization is dominated by distribution shift.

Where This Fits In The Course

Geospatial data	Deep learning foundations	Self-supervised learning	Earth embeddings	Agents and projects

Today gives the machinery. Later lectures ask how to use this machinery to learn reusable geospatial representations.

Bridge To Foundation Models

Foundation models combine today’s ingredients at larger scale.

large data	large architectures	pretraining objectives

The next lecture focuses on self-supervised and multimodal objectives.

Next Lecture

Self-Supervised, Multimodal & Foundation Models

We will ask:

How can models learn from unlabelled Earth observation archives?

How do representations transfer across tasks, regions, and sensors?