Session 1: Semantic Segmentation with U-Net

Advanced Deep Learning for Pixel-Level Analysis

Stylianos Kotsopoulos
EU-Philippines CoPhil Programme

Session Overview

Duration: 1.5 hours **Type:** Theory + Discussion **Goal:** Master semantic segmentation and U-Net for Earth Observation

You will learn:

- What semantic segmentation is and why it matters for EO
- U-Net architecture: encoder, decoder, skip connections
- Loss functions for segmentation: CE, Dice, IoU, Combined
- Real-world EO applications: floods, land cover, roads, buildings
- How to choose the right loss function for your task

Prerequisites:

- Day 2 Session 3 (CNNs)
- Basic Python/TensorFlow
- Understanding of convolution, pooling, padding

Resources:

- Hands-on lab in Session 2 (flood mapping)
- Code examples in notebooks

Part 1: Semantic Segmentation

What is Semantic Segmentation?

Semantic segmentation = classifying every pixel in an image into a category

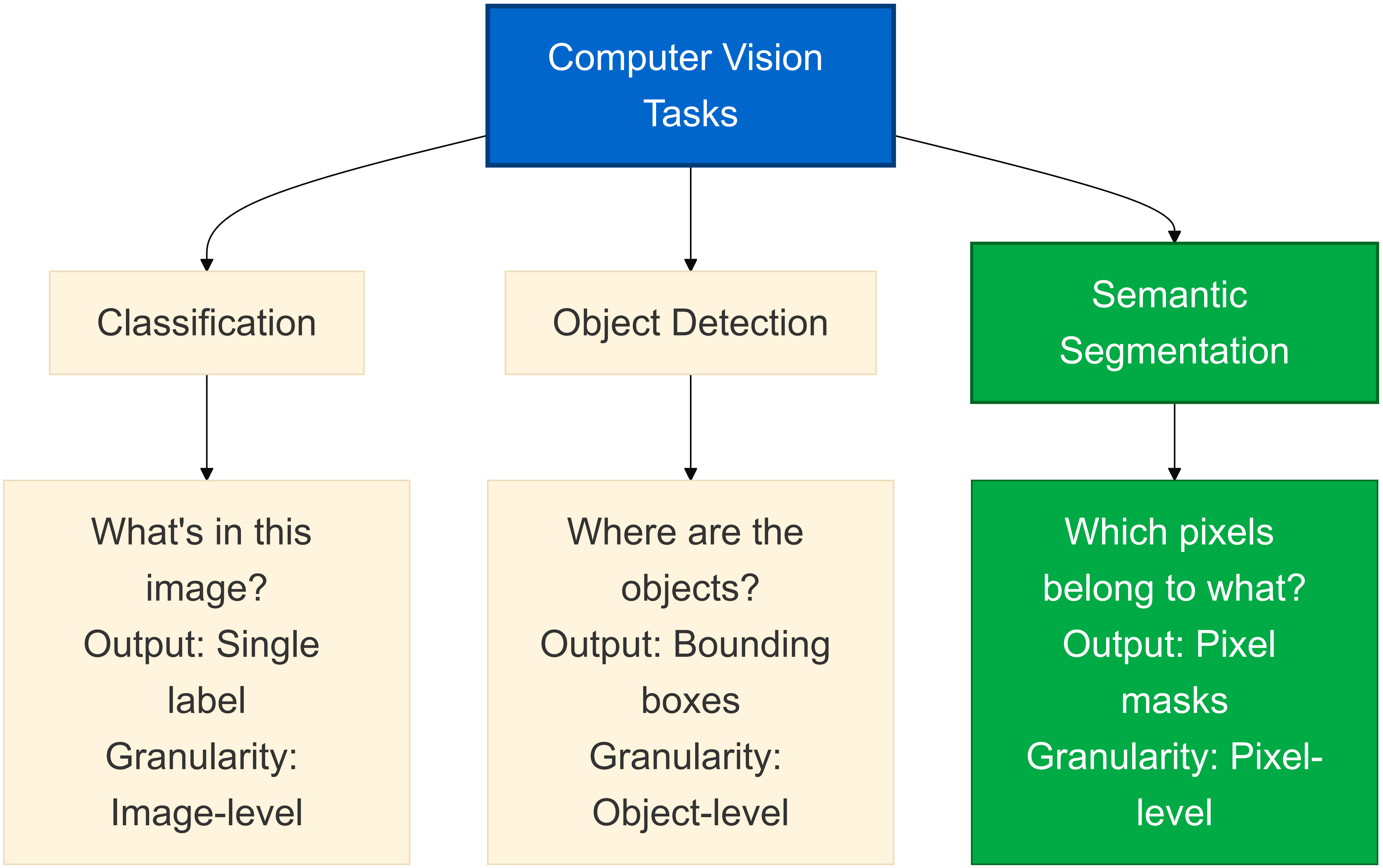
- Unlike **classification**: assigns one label per entire image
- Unlike **object detection**: locates objects with bounding boxes
- Segmentation provides **pixel-wise** detailed map of image content

Why it matters for EO

Pixel-level precision enables:

- Exact boundary delineation (flood edges, forest boundaries)
- Accurate area calculations (km^2 flooded, hectares deforested)
- Detailed change detection over time
- Thematic mapping for decision support

Computer Vision Task Hierarchy



Task Comparison

Aspect	Classification	Object Detection	Semantic Segmentation
Question	What's in this image?	Where are objects?	Which pixels are what?
Output	Single label	Bounding boxes + labels	Pixel-wise mask
Granularity	Image-level	Object-level	Pixel-level
Spatial Info	None	Approximate (boxes)	Precise (pixels)
Computation	Fast	Moderate	Intensive
Use Case	"Contains buildings"	"10 buildings detected"	"Building footprints mapped"

Example: Satellite Image Analysis

Classification

Input: 256×256 satellite patch **Output:** “Urban area”

Info: General category only **Precision:** Image-level

Object Detection

Input: Satellite scene **Output:** 15 bounding boxes

Info: Approximate locations **Precision:** Object-level

Segmentation

Input: Satellite image **Output:** Pixel map (water/building/vegetation/road)

Info: Exact boundaries **Precision:** Pixel-level

> **Important**
>
> For EO applications requiring **precise spatial analysis**, segmentation is essential!

Why Semantic Segmentation for EO?

Advantages

Precise Delineation

- Exact feature boundaries
- Flood extent edges
- Forest-urban transitions
- Agricultural field outlines

Quantitative Analysis

- Accurate area measurements
- Pixel-level precision
- Statistical analysis

Change Detection

- Pixel-by-pixel comparison
- Temporal analysis
- Deforestation tracking
- Urban expansion monitoring

Decision Support

- Disaster response planning
- Targeted relief operations
- Infrastructure planning
- Risk assessment

Part 2: U-Net Architecture

Introduction to U-Net

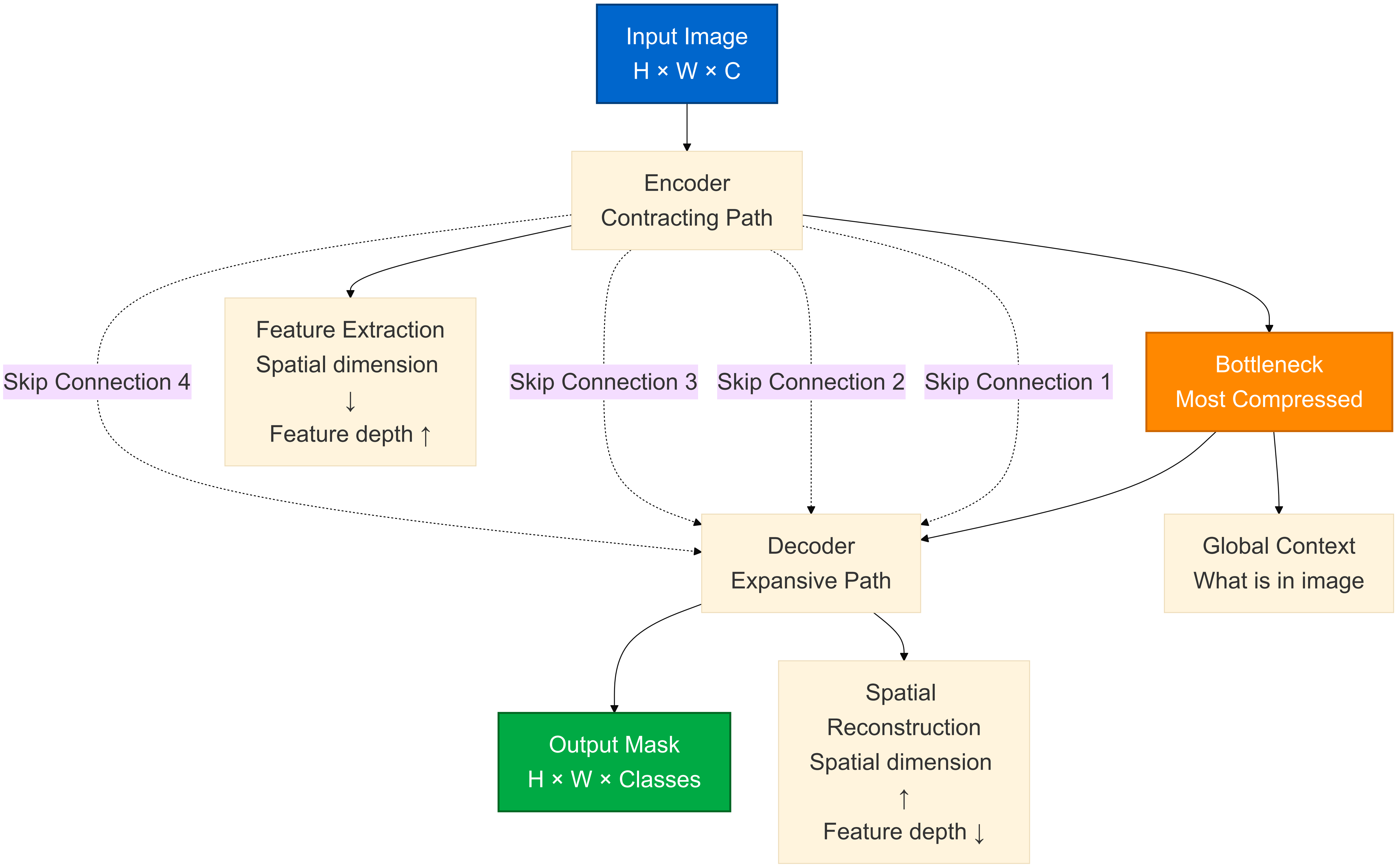
Developed by: Ronneberger et al. (2015) for biomedical image segmentation

Now: One of the most popular architectures for Earth Observation

- **Why “U-Net”?** Architecture shape resembles the letter “U”
- **Structure:** Symmetric encoder-decoder design
- **Key Innovation:** Skip connections that preserve spatial information
- **Strength:** Works well with limited training data

> **Proven Track Record**
>
> U-Net achieves high accuracy in EO applications with just **hundreds to thousands** of training samples (vs millions for other deep learning approaches)

U-Net High-Level Architecture



The Three Components

1. Encoder

Purpose: Extract features

- Conv + ReLU blocks
- MaxPooling (2×2)
- Resolution ↓
- Channels ↑
- Multi-scale features

2. Bottleneck

Purpose: Global context

- Smallest spatial size
- Most feature channels
- Compressed representation
- Semantic understanding
- "What's in the image"

3. Decoder

Purpose: Reconstruct

- Upsampling layers
- Skip concatenation
- Conv blocks
- Resolution ↑
- Channels ↓
- "Where things are"

Encoder (Contracting Path)

Purpose: Extract hierarchical features at multiple scales while compressing spatial information

1. **Convolution Blocks:**
    - Two 3×3 convolutional layers
    - ReLU activation functions
    - (Optional) Batch normalization
    - "Same" padding to preserve dimensions
2. **Downsampling:**
    - 2×2 max pooling
    - Spatial dimensions **halve**
    - Feature channels **double**
    - Creates hierarchical representation

Encoder Example Progression

```
Input:      256×256×3    # RGB satellite image
↓ Conv Block 1
Block 1:    256×256×64   # After convolutions
↓ MaxPool (2×2)
Pool 1:     128×128×64   # Spatial dims halved
↓ Conv Block 2
Block 2:    128×128×128  # More channels
↓ MaxPool (2×2)
Pool 2:     64×64×128
↓ Conv Block 3
Block 3:    64×64×256
↓ MaxPool (2×2)
Pool 3:     32×32×256
↓ Conv Block 4
Block 4:    32×32×512
↓ MaxPool (2×2)
Pool 4:     16×16×512    # Ready for bottleneck
```

Multi-Scale Feature Learning

Early Encoder Layers

Capture fine details:

- Edges and boundaries
- Textures and patterns
- Small features (boats, cars)
- Local context
- High spatial resolution

Deep Encoder Layers

Capture semantics:

- Water bodies
- Urban areas
- Forests
- Agricultural fields
- Global context
- Low spatial resolution

Connection to Day 2

This is the same **CNN hierarchy** you learned! Early layers = low-level features, deep layers = high-level semantic features.

Bottleneck Layer

The central part of U-Net (bottom of the "U")

- **Smallest spatial dimensions** (e.g., 16×16 pixels)
- **Largest feature channels** (e.g., 1024 channels)
- **Maximum context, minimum spatial detail**
- Highly compressed representation

What it captures:

- **Semantic understanding:** "There is water, buildings, vegetation"
- **Global context:** What's in the image overall
- **Lost information:** Precise spatial locations (where exactly)

> **Note**
>
> This trade-off is intentional! The decoder will reconstruct precise locations using skip connections.

Decoder (Expansive Path)

Purpose: Reconstruct spatial resolution to create precise pixel-wise predictions

1. **Upsampling:**
   - Transpose convolution (learnable) OR
   - Bilinear/nearest interpolation + conv
   - Doubles spatial dimensions
   - Halves feature channels
   - Mirrors encoder in reverse
2. **Skip Connection Concatenation:**
   - Copy encoder features
   - Concatenate with decoder features
   - Fuse spatial details + semantics
   - Feature maps must align (same size)

Decoder (Continued)

3. **Convolution Blocks:**
    - Two 3×3 convolutional layers
    - ReLU activation
    - Refine combined features
    - Sharpen predictions
4. **Final Layer:**
    - 1×1 convolution
    - Produces class logits
    - One channel per class
    - Full resolution output

Decoder Example Progression

```
Bottleneck:  16×16×1024   # Most compressed
↓ Upsample (TransposeConv 2×2)
Upsample 1:  32×32×512    # Spatial doubled
↓ Concatenate with encoder Block 4 (32×32×512)
Concat:      32×32×1024   # 512 + 512 channels
↓ Conv Block
Conv Block:  32×32×512    # Refined features
↓ Upsample (TransposeConv 2×2)
Upsample 2:  64×64×256
↓ Concatenate with encoder Block 3 (64×64×256)
Concat:      64×64×512    # 256 + 256 channels
↓ Conv Block
Conv Block:  64×64×256
...
↓ Final Conv 1×1
Final:       256×256×num_classes  # Full resolution!
```

Skip Connections: The Key Innovation

Why Skip Connections Matter

The Problem: Without skip connections, spatial information is lost during downsampling (pooling). The decoder would have to reconstruct precise boundaries from coarse bottleneck features only → **blurry boundaries**.

The Solution: Skip connections preserve edges and small structures by copying high-resolution encoder features directly to the decoder.

Result: Best of both worlds:

- **Encoder:** Captures “what” is in the image (semantic context)
- **Decoder + Skips:** Ensures we know “where” things are (precise localization)

How Skip Connections Work

1. **Encoder** produces feature map: 128×128×64
2. Feature map is **copied and saved**
3. Encoder continues downsampling (pooling)
4. Process continues through **bottleneck**
5. **Decoder** upsamples to 128×128×32
6. **Concatenation:** Decoder (128×128×32) + Encoder copy (128×128×64)
7. **Result:** Combined feature map (128×128×96) with:
   - High-level semantic context from decoder
   - Fine spatial details from encoder

Skip Connections Impact

Real-World Example: Flood Mapping

Without Skip Connections

- Flood boundary accuracy: ±10 pixels
- At 10m resolution: ±100-200 meters
- Blurry edges
- Loss of small features
- Poor for decision-making

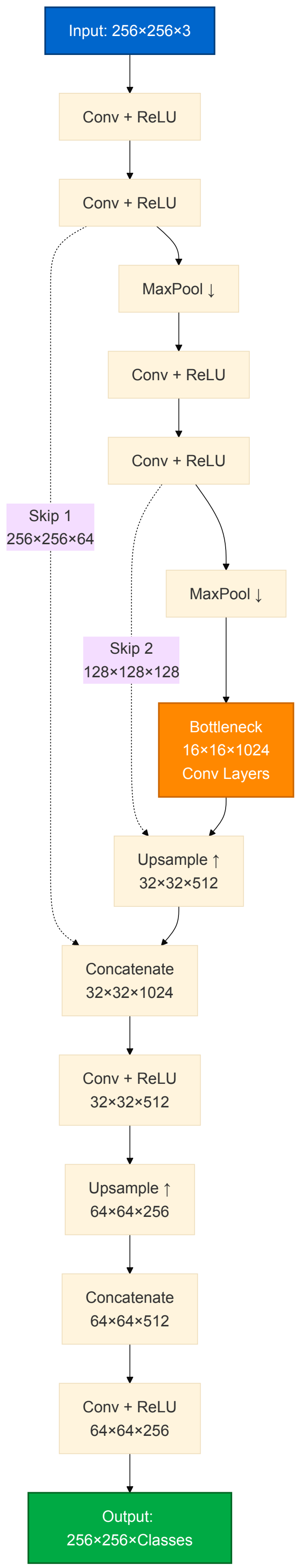
With Skip Connections

- Flood boundary accuracy: ±1-2 pixels
- At 10m resolution: ±10-20 meters
- Sharp boundaries
- Preserves details
- Critical for response planning

. . .

> **Tip**
>
> This precision is **critical** for applications requiring legal boundaries, property lines, or hazard zone delineation!

U-Net Complete Information Flow



Part 3: Applications in EO

Why U-Net is Popular in EO

Data Efficiency

- Works with **hundreds to thousands** of samples
- (vs millions for other DL methods)
- Critical when labeled EO data is expensive
- Data augmentation helps further
- Expert annotation is time-consuming

Multi-Scale Learning

- Hierarchical structure
- Captures local textures (early layers)
- Captures global context (deep layers)
- Handles varied EO feature scales
- Small boats to large water bodies

Spatial Precision

- Skip connections preserve fine boundaries
- Millimeter to meter level accuracy
- Essential for legal boundaries
- Property lines
- Hazard zone delineation

Transfer Learning

- Encoder can use pre-trained weights
- ImageNet → Satellite imagery
- Domain adaptation
- Improves limited-data performance
- Fine-tune to specific RS tasks

Application 1: Flood Mapping

Use Case

Disaster response and damage assessment

Data Sources:

- Sentinel-1 SAR (cloud-penetrating)
- Sentinel-2 optical (when clear)

Task:

- Binary segmentation: Flooded / Non-flooded
- Input: SAR backscatter (VV, VH) or RGB+NIR
- Output: Precise flood extent mask

Why U-Net Excels

- High accuracy in delineating water
- Captures flood patterns in SAR
- Rapid assessment (hours after acquisition)
- Robust with small training datasets
- Pixel-wise flood/non-flood classification

Philippine Example

Typhoon Ulysses (2020)

- Central Luzon floods
- U-Net on Sentinel-1 data
- Precise inundation extent
- Pampanga River Basin mapping

Flood Mapping Benefits

- **Rapid mapping** within hours of satellite acquisition
- **Precise area calculations** for damage assessment
- **Time-series monitoring** of flood evolution and recession
- **GIS integration** for evacuation planning
- **Relief distribution** planning
- **Emergency response** coordination

Session 2 Preview

Tomorrow’s hands-on lab: Train U-Net for Central Luzon flood mapping using Sentinel-1 SAR data!

Application 2: Land Cover Mapping

Use Case

Environmental monitoring, urban planning, biodiversity assessment

Data Sources:

- Sentinel-2 multispectral (10m)
- Landsat 8/9 (30m, long time series)
- High-resolution commercial imagery

Task:

- Multi-class: Water, Forest, Urban, Agriculture, Barren, Mangrove
- Input: Multi-spectral bands (RGB, NIR, SWIR, Red Edge)
- Output: Detailed land cover map

U-Net's Strength

- Broad context + precise boundaries
- Exact delineation between classes
- Outperforms pixel-based methods
- Outperforms patch-based methods
- Research-proven accuracy

Benefits

- Pixel-accurate thematic maps
- Change detection (deforestation, urbanization)
- Biodiversity habitat assessment
- Carbon stock estimation
- Climate reporting

Application 3: Road Network Extraction

Challenges

- Thin linear features (difficult to detect)
- Tree and shadow occlusion
- Complex urban backgrounds
- Need continuous structure (no gaps)

. . .

U-Net Advantages

- Skip connections preserve **road continuity**
- Learns **linear patterns** across image
- Handles varying widths (highways → small paths)
- Can trace continuous structures

Road Extraction Details

Task:

- Binary segmentation: Road / Background
- Input: High-resolution aerial/satellite RGB or SAR
- Output: Road network mask for vectorization

Applications:

- Automated map updating (rural areas)
- Transportation network planning
- Accessibility analysis
- Disaster response routing

Application 4: Building Footprint Delineation

Use Case

Urban mapping, population estimation, disaster risk assessment

Philippine Context:

- Monitor unplanned urban growth (Metro Manila)
- Identify disaster-vulnerable communities
- Support urban planning and housing programs
- Informal settlement detection

Task Details

- Binary/multi-class: Building / Background
- Input: Very high-resolution (<1m) or Sentinel-2
- Output: Building footprint polygons

U-Net Performance

- Outlines individual buildings
- Maps dense informal settlements
- Handles complex backgrounds
- Variants: Residual U-Net, Attention U-Net
- Core: encoder-decoder + skip connections

Building Footprint Benefits

- **Automated mapping at scale**
- **Pre/post disaster damage assessment**
- **3D city model generation** (with height data)
- **Infrastructure planning**
- **Risk assessment**
- **Population estimation**
- **Urban growth monitoring**

Application 5: Vegetation & Crop Monitoring

Use Case

Precision agriculture, forestry, ecosystem health

Data Sources:

- Sentinel-2 (5-day revisit)
- PlanetScope (3m daily)
- UAV imagery (field-scale)

Task:

- Multi-class: Rice, corn, sugarcane, coconut
- Or binary: Vegetation / Non-vegetation
- Input: Multi-temporal + multi-spectral
- Output: Crop type map or vegetation mask

U-Net Applications

- Identify crop fields at pixel level
- Monitor forest cover
- Track agricultural areas (food security)
- Tree cover for forestry management
- Detect vegetation changes

Benefits

- Yield prediction
- Harvest planning
- Irrigation monitoring
- Early disease detection
- Deforestation tracking
- Illegal logging detection

Research Evidence

Proven Performance

Across all these applications, research shows:

- **High segmentation accuracy** in remote sensing
- **Robust results with small training datasets**
- **Efficiency of architecture** (medical imaging success)
- **Outperforms traditional methods** (pixel-based, patch-based)
- **Wide adoption** across EO community

Part 4: Loss Functions

Why Loss Functions Matter

Loss function = mathematical measure comparing predicted mask to ground truth

- Tells the model “how wrong” its predictions are
- Guides weight updates during training
- **Critical choice** affecting model behavior

The Segmentation Challenge

Unlike classification (one value), segmentation compares **entire images** pixel-by-pixel:

- Potentially **millions** of pixel predictions
- Different loss functions emphasize different aspects
- Pixel-wise accuracy vs region overlap vs boundary precision

Loss Function Emphasis

Pixel-wise

Focus: Individual pixel correctness

Example: Cross-Entropy

Asks: Is each pixel correct?

Region Overlap

Focus: Spatial agreement

Example: Dice, IoU

Asks: Does predicted region match true region?

Boundary Accuracy

Focus: Edge precision

Example: IoU, Boundary Loss

Asks: Are edges precisely delineated?

Key Point

The choice of loss function **critically affects model behavior**. Poor choice → useless predictions!

Challenge: Class Imbalance in EO

Common Imbalanced Scenarios

Flood Mapping

- 95% non-flooded pixels
- 5% flooded pixels

Ship Detection

- 99.5% water/land
- 0.5% ships

Building Segmentation

- 80% background
- 20% buildings

Problem with Simple Accuracy

```
# Model predicts: ALL pixels = "non-flooded"
# Accuracy: 95% ✓ (looks great!)
# But: Completely useless - missed all floods!
```

Why? Vanilla cross-entropy is dominated by majority class.

Result: Model predicts majority class for all pixels → high accuracy but **no useful information**!

What We Need from Loss Functions

For imbalanced EO data, loss functions must:

1. **Handle severe class imbalance**
2. **Focus on minority (critical) class**
3. **Reward region overlap**, not just pixel-wise correctness
4. **Ensure accurate boundaries** (flood edges, building outlines)
5. **Provide stable gradients** for training

Loss Function 1: Pixel-wise Cross-Entropy

How it Works

- Treat each pixel as **independent classification**
- Compare predicted probability to true class
- Negative log-likelihood
- Average loss across all pixels

Formula

$$\text{Cross-Entropy} = -\sum_{i=1}^{N} y_{\text{true},i} \cdot \log(y_{\text{pred},i})$$

Where N = total number of pixels

Cross-Entropy: Pros & Cons

Advantages ✓

- Standard, well-understood
- Strong, stable gradients
- Works with multi-class (softmax)
- Effective and straightforward
- Well-supported in frameworks

Disadvantages ✗

- **Dominated by majority class**
- Doesn't optimize spatial overlap
- Can ignore minority classes
- Pixel-level, not region-level
- Needs balancing for EO

When to use: Balanced datasets (~50/50 distribution) or **with class weighting**

Weighted Cross-Entropy

Solution to Imbalance

Assign **higher weight** to under-represented classes

Formula

$$\text{Weighted CE} = -\sum_{i=1}^{N} w_{\text{class}} \cdot y_{\text{true},i} \cdot \log(y_{\text{pred},i})$$

Example

- 95% background pixels → weight = 1.0
- 5% flood pixels → weight = **19.0** (inverse frequency: 95/5)

Effect

- Model pays **19× more attention** to flood pixels
- Heavily penalized for missing floods
- Common remedy for imbalance

Weighted CE Implementation

```python
# TensorFlow/Keras example
loss = tf.keras.losses.CategoricalCrossentropy(
    class_weight={
        0: 1.0,    # background
        1: 19.0    # flood (95/5 ratio)
    }
)

# Compile model
model.compile(
    optimizer='adam',
    loss=loss,
    metrics=['accuracy', 'IoU']
)
```

Note

Weighted CE provides strong gradients while addressing imbalance, but still focuses on **pixel-wise accuracy** (not region overlap).

Loss Function 2: Dice Loss

Concept

Measure **overlap** between prediction and ground truth regions

Formula

$$\text{Dice Coefficient} = \frac{2 \times |P \cap T|}{|P| + |T|}$$

$$\text{Dice Loss} = 1 - \text{Dice Coefficient}$$

Where:

- P = predicted foreground pixels
- T = true foreground pixels
- $\cap$ = intersection (overlap)

Dice Loss Interpretation

Dice Scores

- **Dice = 1.0:** Perfect overlap
- **Dice = 0.5:** 50% overlap
- **Dice = 0.0:** No overlap

Loss Values

- **Loss = 0.0:** Perfect (lower is better)
- **Loss = 0.5:** Moderate error
- **Loss = 1.0:** Complete failure

Visual Example

```
Predicted:    ██░░
True:         ░███
Intersection: █
Dice = 2×2/(4+6) = 0.40
Loss = 1 − 0.40 = 0.60
```

Better:

```
Predicted:    ░███
True:         ░███
Intersection: ███
Dice = 2×6/(6+6) = 1.00
Loss = 0.00 ✓
```

Why Dice for Imbalanced Data?

- Focuses on **relative overlap** of object (minority class)
- Treats foreground and background **asymmetrically**
- Small flooded patch contributes as much as large non-flood region
- **Inherently handles class imbalance** (no manual weighting needed)
- Well-suited when target objects occupy **small fraction** of image
- Doesn’t let model predict only majority class
- **Equal weight** to false positives and false negatives

Dice Loss: Pros & Cons

Advantages ✓

- **Inherently robust to class imbalance**
- Directly optimizes overlap metric (F1)
- Excellent for small objects
- No manual class weights needed
- Prevents ignoring minority classes
- Medical imaging proven

Disadvantages ✗

- Less stable gradients (noisy early training)
- May converge slower than CE
- Requires careful implementation (avoid division by zero)
- Can be sensitive to initialization

Medical Imaging Parallel

Used to segment **tumors** (tiny area) in medical images—same challenge as small flood patches in vast satellite scenes!

Loss Function 3: IoU Loss (Jaccard Index)

Concept

Similar to Dice, directly optimizes the **IoU metric**

Formula

$$\text{IoU} = \frac{|P \cap T|}{|P \cup T|}$$

$$\text{IoU Loss} = 1 - \text{IoU}$$

Where $\cup$ = union of predicted and true pixels

Dice vs IoU

Mathematical Relationship

$$\text{Dice} = \frac{2 \times \text{IoU}}{1 + \text{IoU}}$$

Dice

Formula: $\frac{2 \times \text{Intersection}}{\text{Sum of areas}}$

- More forgiving (2× numerator)
- Smoother gradients
- More stable training
- Common in medical/EO training

IoU

Formula: $\frac{\text{Intersection}}{\text{Union}}$

- Stricter evaluation
- Can be less stable
- Standard in challenges
- Higher boundary emphasis

IoU Properties

- **Robust to class imbalance** (like Dice)
- Emphasizes **boundary accuracy**
- Penalizes false positives and false negatives **equally at region level**
- Standard metric in segmentation challenges
- Useful when **boundary delineation is crucial** (geographic mapping)

Practical Guidance

In practice, both Dice and IoU work well for imbalanced EO data. Try both on your dataset—difference is often small. Dice is slightly more popular in training (smoother), IoU common for evaluation.

Loss Function 4: Combined Losses

Best of Both Worlds

Combine complementary loss functions:

$$\text{Total Loss} = \alpha \cdot \text{CE Loss} + \beta \cdot \text{Dice Loss}$$

Where α and β are weights (e.g., $\alpha = 0.5$, $\beta = 0.5$)

Why Combine Losses?

Cross-Entropy Provides

- Strong, stable gradients
- Pixel-wise correctness
- Well-understood training dynamics
- Fast convergence

Dice Provides

- Overlap focus
- Handles imbalance
- Region-level accuracy
- Minority class emphasis

Combined Benefits

- **Stable training** from CE's strong signal
- **Balanced optimization** from Dice's overlap focus
- Often achieves **best results in practice**
- Improves both per-pixel AND region accuracy

Combined Loss Implementation

```python
def combined_loss(y_true, y_pred):
    # Cross-entropy component
    ce = tf.keras.losses.categorical_crossentropy(
        y_true, y_pred
    )

    # Dice loss component
    dice = dice_loss(y_true, y_pred)

    # Combine with equal weights
    return 0.5 * ce + 0.5 * dice

# Use in model
model.compile(
    optimizer='adam',
    loss=combined_loss,
    metrics=['accuracy', dice_coefficient, 'IoU']
)
```

Other Advanced Losses

Additional Options

Focal Loss

- Modified cross-entropy
- Down-weights easy/background examples
- For **extremely imbalanced** cases
- More common in object detection
- Can help with hard examples

Boundary Loss

- Explicitly penalizes boundary errors
- For applications requiring precise edges
- Can combine with Dice/CE

Tversky Loss

- Generalization of Dice
- Adjustable FP/FN weighting
- Fine-tune precision vs recall trade-off

Loss Function Selection Guide

Decision Framework

1. **Is data balanced?** (~50/50)
   - **Yes** → Standard Cross-Entropy
   - **No** → Continue
2. **Is minority class critical?** (floods, damage, ships)
   - **Yes** → Dice or IoU Loss
   - **Somewhat** → Weighted Cross-Entropy
3. **Need most stable training?**
   - **Yes** → Combined Loss (CE + Dice)
   - **No** → Pure Dice/IoU is fine
4. **Is boundary accuracy critical?**
   - **Yes** → IoU Loss or Combined
   - **Moderate** → Dice is sufficient

EO Common Practice

Application	Recommended Loss	Reason
Flood mapping	Dice or Combined	Severe imbalance, critical boundaries
Balanced land cover	Cross-Entropy	Classes relatively balanced
Building extraction	Dice or IoU	Precise footprints matter
Road extraction	Combined Loss	Thin features, need continuity
Ship detection	Dice	Extreme imbalance, small objects
Crop classification	Weighted CE or Combined	Moderate imbalance

Practical Example: Flood Mapping

Scenario:

- Dataset: 1000 Sentinel-1 SAR images (Central Luzon)
- Class distribution: 92% non-flooded, 8% flooded
- Goal: Precise flood extent for disaster response

Experiment Results

Loss Function	IoU Score	Notes
Cross-Entropy	0.12	Predicts mostly non-flooded; **trivial solution** ✗
Weighted CE	0.54	Better; weight=11.5×; some false positives
Dice Loss	0.68	Good recall; slightly noisy; handles imbalance ✓
Combined (CE + Dice)	**0.73**	**Best balance; stable training** ✓✓

Key Insight from Example

Loss Choice is Critical!

For flood mapping (binary segmentation, severe imbalance):

- **Poor choice** (vanilla CE): IoU = 0.12 (useless!)
- **Good choice** (Combined): IoU = 0.73 (excellent!)

Difference: Trivial predictions vs. operationally useful model

The right loss pushes the model to correctly segment the minority class rather than achieving high but **meaningless accuracy**.

Key Takeaways

Session 1 Summary: Segmentation

Semantic Segmentation

✓ **Pixel-wise classification** providing precise boundaries ✓ Differs from classification (labels) and detection (boxes) ✓ Essential for EO: exact spatial extent, area calculations ✓ Enables detailed thematic mapping and spatial analysis ✓ **Critical** for disaster response, planning, monitoring

Session 1 Summary: U-Net

U-Net Architecture

✓ **Encoder-decoder** structure with U-shape ✓ **Skip connections** = key innovation (preserve spatial details) ✓ Combines "what" (semantic context) + "where" (localization) ✓ Works with **limited data** (hundreds to thousands of samples) ✓ **Widely adopted** across EO community ✓ Uses same CNN building blocks from Day 2 ✓ Proven high accuracy in RS applications

Session 1 Summary: Loss Functions

Loss Functions

✓ **Cross-Entropy:** Standard, strong gradients, but imbalance-sensitive ✓ **Weighted CE:** Addresses imbalance via class weighting ✓ **Dice/IoU:** Inherently handle imbalance, optimize region overlap ✓ **Combined losses** often best in EO (e.g., CE + Dice) ✓ **Choice critically impacts** model behavior ✓ Consider: data balance, boundary importance, object size ✓ **Can mean difference** between useless and excellent results

Session 1 Summary: Applications

EO Applications

✓ Flood mapping, land cover, buildings, roads, vegetation ✓ High accuracy even with **small datasets** ✓ **Outperforms** older pixel-based and patch-based methods ✓ Wide adoption across EO community ✓ Proven results in Philippine contexts (Typhoon Ulysses, urban mapping) ✓ From disaster response to urban planning to agriculture

Golden Rule

Think About Your Problem!

Don't just accept the default loss function.

Think about:

- Class distribution (balanced or imbalanced?)
- What matters most (boundaries, overlap, pixel accuracy?)
- Is minority class critical?
- What are you optimizing for?

Then pick (or tune) your loss accordingly for the best results!

Preparation for Session 2

Next: Hands-on Flood Mapping Lab

Session 2 Preview

What You'll Do:

1. Load Sentinel-1 SAR data (Typhoon Ulysses, Central Luzon)
2. Build U-Net model in TensorFlow/Keras
3. Train with **Dice Loss** (or combined loss)
4. Evaluate performance: IoU, F1-score, precision, recall
5. Visualize flood predictions and create export maps

Dataset:

- ~500-1000 pre-processed SAR patches (256×256 pixels)
- Binary flood masks (flooded / non-flooded)
- Real flood event from major Philippine river basin

Expected Results & Preparation

Expected Results

- **IoU > 0.70** with properly trained model
- Visual flood extent maps ready for GIS integration
- Understanding of full U-Net training pipeline

To Prepare

- Ensure **Google Colab access**
- Check **GPU availability** (Runtime → Change runtime type → GPU)
- Review Python and NumPy basics if needed
- Have **patience** - model training takes time!
- Bring questions from today's session!

Resources

Core References

Foundational Paper:

- Ronneberger et al. (2015). U-Net: Convolutional Networks for Biomedical Image Segmentation. arXiv:1505.04597

Practice Datasets:

- Sen1Floods11 (Global flood dataset, Sentinel-1)
- DeepGlobe Land Cover Challenge
- SpaceNet Building Detection
- Landcover.ai (High-res orthophotos)

Discussion Questions

Before Session 2, consider:

1. **What EO applications in your work** could benefit from semantic segmentation vs classification/detection?
2. **How would you validate** segmentation results in the field for disaster response?
3. **What challenges** do you anticipate with limited training data for Philippine contexts?
4. **How might transfer learning** from other regions help Philippine EO applications?

Thank You!

Questions?

Stylianos Kotsopoulos EU-Philippines CoPhil Programme

Session 2: Flood Mapping Lab (Hands-on with U-Net)

- Session materials online
- Jupyter notebooks for practice
- Links to datasets and tutorials
- PhilSA and DOST-ASTI resources

This session is part of the CoPhil 4-Day Advanced Training on AI/ML for Earth Observation, funded by the European Union under the Global Gateway initiative and delivered in partnership with PhilSA and DOST.