02_Validation_Dataset_Insights

02 — Validation Dataset Insights

Folder Location

```
DATASET/Offroad_Segmentation_Training_Dataset/val/
├── Color_Images/    (289 PNG files)
└── Segmentation/    (289 PNG files)
```

What's Inside

Property	Details
Total Image Pairs	**289** (RGB + mask)
Image Format	`.png`
Naming Convention	`cc0000016.png`, `cc0000019.png`, etc. — same `cc` prefix style as training, but **different IDs** (no overlap with train)
Color Images	Synthetic desert RGB scenes (same style as training)
Segmentation Masks	Identical format to training — raw pixel values that get remapped to 0–9
Image-Mask Pairing	Same filename in both subfolders

Validation vs Training — Key Differences

Aspect	Training Set	Validation Set
Image Count	293	289
Split Ratio	~50.3%	~49.7%
Image IDs	`cc0000012` – `cc0000XXX` (one range)	`cc0000016`, `cc0000019` ... (different IDs, non-overlapping)
Purpose	Model learns from this	Model is evaluated on this (unseen during gradient updates)
Same Scene Domain?	Yes — same desert biome	Yes — same desert biome but different specific locations

> **Notable**: The train/val split is nearly 50-50 (~293 vs ~289). This is an **unusually even split** — most real projects use 80/20 or 70/30. This means less training data but stronger validation signal.

How Validation is Used in Training (`train_segmentation.py`)

During each epoch, the training script:

1. **Trains** on the train set (with gradient updates)
2. **Evaluates** on the val set (no gradients) — computes val loss
3. **Calculates full metrics** on BOTH train and val sets using `evaluate_metrics()`:
   - **Mean IoU** (Intersection over Union) — main hackathon metric (80 points)
   - **Mean Dice Score** (F1 per class)
   - **Pixel Accuracy** (% correct pixels)

Python

```python
# From train_segmentation.py main():
val_iou, val_dice, val_pixel_acc = evaluate_metrics(
    classifier, backbone_model, val_loader, device, num_classes=10
)
```

Metrics Tracked Per Epoch

Metric	Formula	Why It Matters
IoU (Jaccard)	`Intersection / Union` per class, then averaged	Primary hackathon score — measures overlap between prediction and ground truth
Dice (F1)	`2 * Intersection / (Pred + GT)` per class	Alternative overlap metric; more forgiving than IoU
Pixel Accuracy	`Correct pixels / Total pixels`	Simple but misleading if classes are imbalanced (sky/landscape dominate)

> **Hackathon scoring**: IoU = **80 points** out of 100. This is THE metric to optimize.

Key Observations

1. **Validation set has ground truth masks** — this means you can fully evaluate locally before submitting
2. **50/50 split is generous for validation** — consider combining train+val and doing your own 80/20 split, or using k-fold cross-validation for better training
3. **Same domain but different locations** — the val set tests same-biome generalization (desert → different desert area)
4. **No model checkpointing in default script** — the provided `train_segmentation.py` does **not** save the best model (by val IoU). It only saves the final epoch's model. You should add `if val_iou > best_iou: save checkpoint`
5. **Val loss is computed but not used for early stopping** — another improvement opportunity

---

Suggestions for Better Validation

- **Add best-model saving**: Save checkpoint when `val_iou` improves
- **Add early stopping**: Stop if val loss doesn't improve for N epochs
- **Consider re-splitting**: Merge train+val (582 total) → re-split 80/20 (466 train, 116 val) for more training data
- **Watch for overfitting**: With only 293 training images and no augmentations, the model will likely overfit fast