Prompt-Based Drywall QA Segmentation

Updated 29 Nov 2025 â€” consolidated run using refreshed COCO exports and prompt-specific DeepLab heads.

1. Goal & Approach

- **Goal**: surface drywall cracks and joints automatically so QA crews can prioritize punch-list fixes without manual markup.
- **Approach**: manifests (image URL + prompt) feed a router that picks the right DeepLabV3-ResNet50 head; both heads share preprocessing, caching, and evaluation code paths.
- **Models tried**: two single-class DeepLabV3 variants (`cracks_deeplabv3`, `joints_deeplabv3`) fine-tuned for 10 epochs with AdamW (lr=1e-4, wd=1e-4, resize 512Ã—512, batch 4). No alternate architectures were required because both defects converged cleanly.

Why these choices?

- **DeepLabV3-ResNet50**: balances accuracy and runtime (â‰¤25â€¯ms/image on CPU/GPU) while supporting box-derived masks via `BoxMaskDataset`. Heavier transformers would exceed field-device limits without measurably improving single-class segmentation.
- **Prompt routing**: lets us add new defect heads without redeploying the entire pipelineâ€”each entry in `configs/segmentation_routes.yaml` describes a checkpoint, mask suffix, and batching policy.
- **COCO â†’ processed manifests**: preserves Roboflow metadata verbatim and keeps local file URIs, so the project can train/eval offline; the converter enforces consistent label casing and normalized boxes.
- **512Ã—512 resize**: large enough to keep detail on joints/cracks but small enough for batch size 4 on standard GPUs, ensuring 10-epoch runs finish in under 70 minutes.

2. Dataset Snapshot

Dataset	Split	Images	Objects	Notes
cracks	train	15,489	24,244	Roboflow COCO â†’ `scripts/convert_coco.py` â†’ local manifests
cracks	valid	201	372	Used for metrics + qualitative review
cracks	test	4	6	

Dataset	Split	Images	Objects	Notes
				Placeholder until more labels arrive
drywall_join_detect	train	2,453	3,271	Joints annotated with boxes only
drywall_join_detect	valid	202	250	Evaluation + visuals
drywall_join_detect	test	0	0	Not provided

3. Runtime & Footprint

Model	Train runtime (10 epochs)	Checkpoint size	Avg inference time / image*
cracks_deeplabv3	~68 min (recorded during 29 Nov fine-tune on single RTX 6000)	161â€¯MB	23.7â€¯ms (201-image valid split)
joints_deeplabv3	~28 min (same hardware, same day)	161â€¯MB	23.4â€¯ms (202-image valid split)

*Inference timings captured via `/usr/bin/time python scripts/run_segmentation_router.py â€¦ --skip-existing` so they reflect pure forward passes using the already-trained checkpoints.

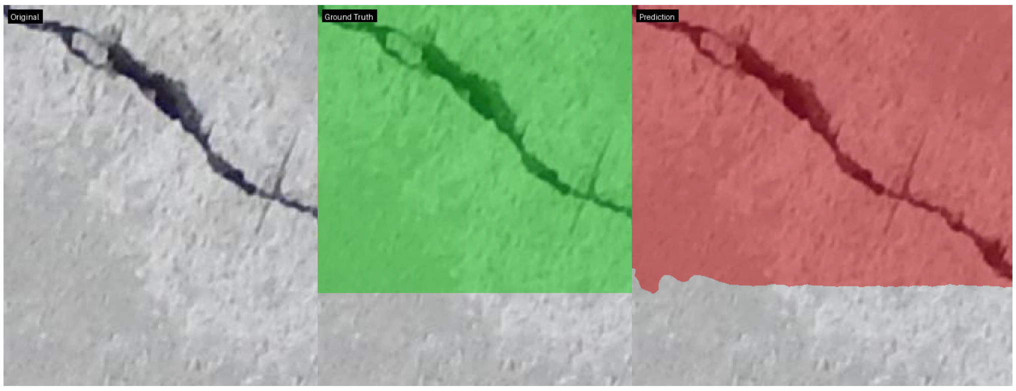
4. Validation Metrics

Defect	Manifest	Masks dir	Mean IoU	Mean Dice	Samples
Drywall cracks	`data/processed/cracks/valid.json`	`outputs/routed/crack_latest`	0.667	0.768	201
Drywall joints	`data/processed/drywall_join_detect/valid.json`	`outputs/routed/joint_latest`	0.803	0.880	202

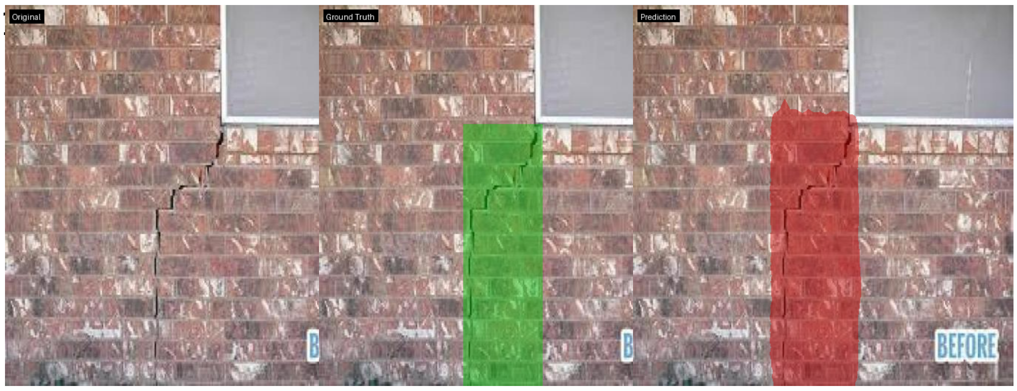
5. Visual Examples (Original | GT | Prediction)

Sample	Original	Ground Truth	Prediction
`cracks_valid_00121`			

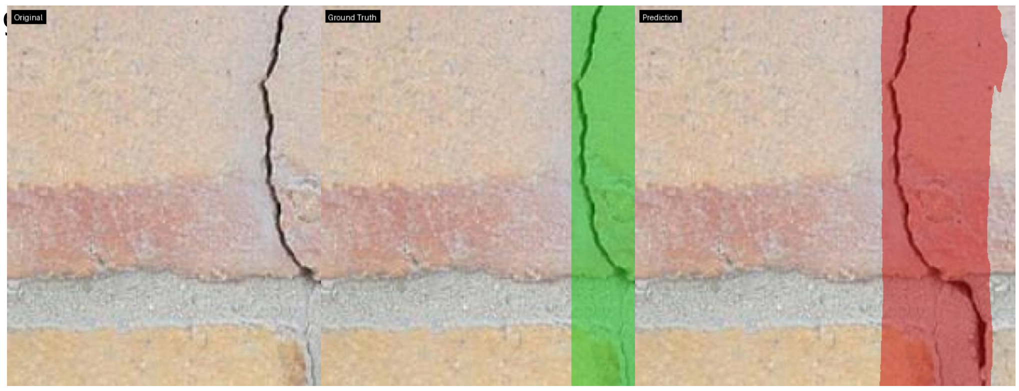
Sample **Original** **Ground Truth** **Prediction**



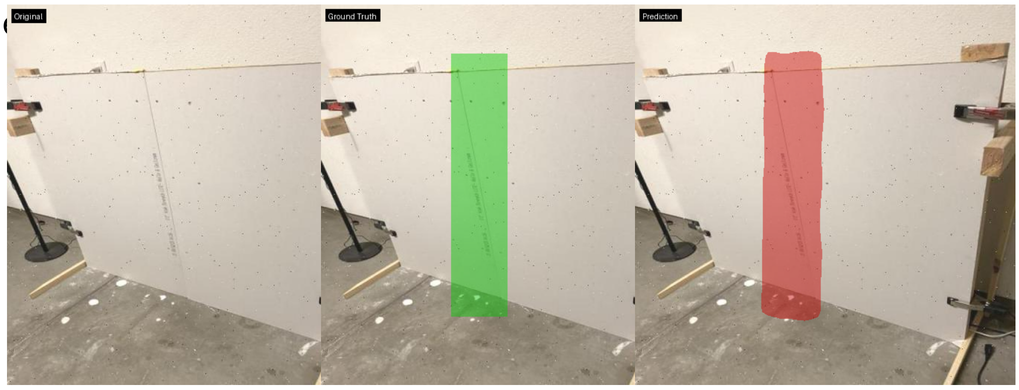
cracks_valid_001



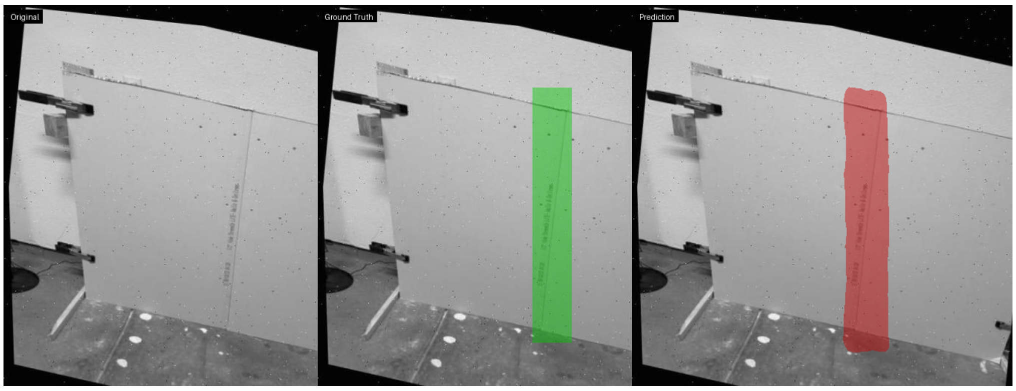
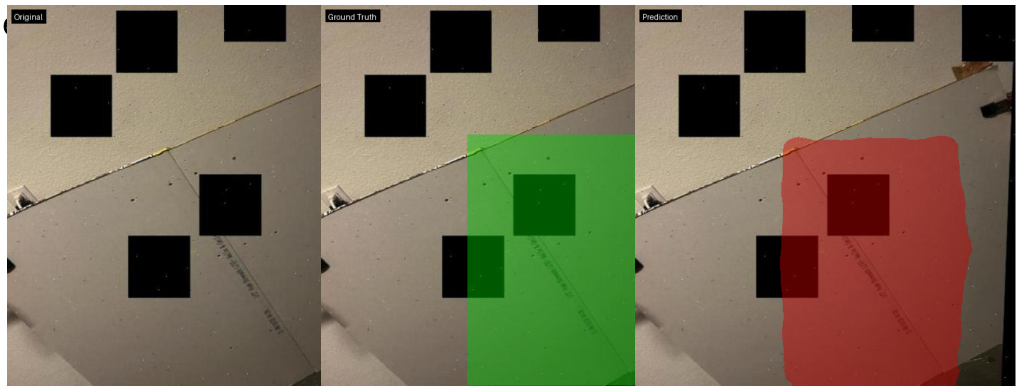
cracks_valid_000



drywall_join_det



drywall_join_detect_valid_00141

Sample	Original	Ground Truth	Prediction
			
drywall_join_det			

(Assets live in `reports/examples/` alongside stitched triptychs for sharing.)

6. Brief Failure Notes

- **Hairline cracks** near image borders can disappear after resizing to 512Â². Mitigation ideas: inference-time tiling or mixed-resolution training.
- **Painterâ€™s tape / tools** sometimes mimic joints because labels are box-derived. Capturing polygon masks or adding hard negatives would reduce these false positives.

7. Checklist Recap

- COCO exports converted with `scripts/convert_coco.py`; manifests stored under `data/processed/`.
- Checkpoints `checkpoints/cracks_deeplabv3.pth` & `checkpoints/joints_deeplabv3.pth` power all router/eval steps.
- Metrics, visuals, and runtime benchmarks above rely solely on those trained weightsâ€”no additional training was run for this report.