Prompted Segmentation for Drywall

Project Report

15th February 2026

1 Goal Summary

The objective of this project is to develop a text-conditioned semantic segmentation model capable of identifying construction defects based on natural language prompts. Specifically, the model segments:

- **Cracks** when given the prompt: "segment crack"
- **Drywall taping areas** when given the prompt: "segment taping area"

This approach enables a single unified model to perform multiple segmentation tasks conditioned on textual descriptions, eliminating the need for separate models per task.

2 Approach

2.1 Model Architecture

We employ a **Text-Conditioned UNet** architecture with Feature-wise Linear Modulation (FiLM) layers to condition image features on text embeddings. The architecture consists of four main components:

1. **Image Encoder**: A 3-level UNet-style encoder with convolutional blocks
   - Layer 1: $3 \rightarrow 32$ channels
   - Layer 2: $32 \rightarrow 64$ channels (after max pooling)
   - Layer 3: $64 \rightarrow 128$ channels (after max pooling)
2. **Text Encoder**: Simple word embedding-based encoder
   - Vocabulary size: 6 tokens (`<PAD>`, `<UNK>`, `segment`, `crack`, `taping`, `area`)
   - Embedding dimension: 128
   - Mean pooling over token embeddings + FC layer
3. **FiLM Layer**: Modulates image features f with text embeddings t

$$\text{FiLM}(f, t) = \gamma(t) \odot f + \beta(t) \tag{1}$$

   where γ and β are learnable affine transformations.
4. **Decoder**: UNet-style decoder with skip connections
   - Transposed convolutions for upsampling
   - Skip connections from encoder features
   - Final 1×1 convolution for binary segmentation

2.2 Training Configuration

Parameter	Value
Input Image Size	256 × 256
Loss Function	Binary Cross-Entropy with Logits
Optimizer	Adam
Learning Rate	1e-3 (reduced to 5e-4 at epoch 19)
Batch Size	16
Epochs	20
Early Stopping Patience	10 epochs
Weight Decay	1e-4
LR Scheduler	ReduceLROnPlateau (patience=3, factor=0.5)
Device	CUDA (GPU)

Table 1: Training hyperparameters

3 Dataset

3.1 Data Sources

The dataset combines two Roboflow datasets in COCO format:

- **Cracks Dataset** (`cracks.v1i.coco`): Images of structural cracks
- **Drywall Taping Dataset** (`Drywall-Join-Detect.v1i.coco`): Images of drywall joint/taping areas

3.2 Data Split

Split	Images	Masks
Training	5,984	5,984
- Cracks	5,164	5,164
- Drywall	820	820
Validation	403	403
- Cracks	201	201
- Drywall	202	202
Total	**6,387**	**6,387**

Table 2: Dataset statistics

3.3 Data Preprocessing

Critical Fix: Roboflow augmentation handling. Each base image has multiple augmented versions (e.g., `00002_jpg.rf.3119f...`, `00002_jpg.rf.be547...`), each with different annotations. We preserve the full filename including the `.rf.XXXXX` identifier to ensure exact image-mask pairing.

Mask generation uses COCO segmentation polygons (not bounding boxes) with `cv2.fillPoly()` for accurate shape representation.

4 Results

4.1 Training Metrics

Metric	Epoch 1	Epoch 20	Improvement
Training Loss	0.2626	0.1331	−49%
Validation Loss	0.2605	0.2057	−21%
Best Val Loss	–	**0.2057**	–

Table 3: Training and validation loss over 20 epochs

Key Observations:

- Good convergence: Both losses decrease steadily over 20 epochs
- Minimal overfitting: Gap between train and val loss is 0.0726 (acceptable)
- Stable learning: Learning rate reduced at epoch 19 (0.001 → 0.0005)
- Best model saved at epoch 20 with validation loss of 0.2057
- Mean IoU: 0.2808 ± 0.1797 (evaluated on 200 validation samples)

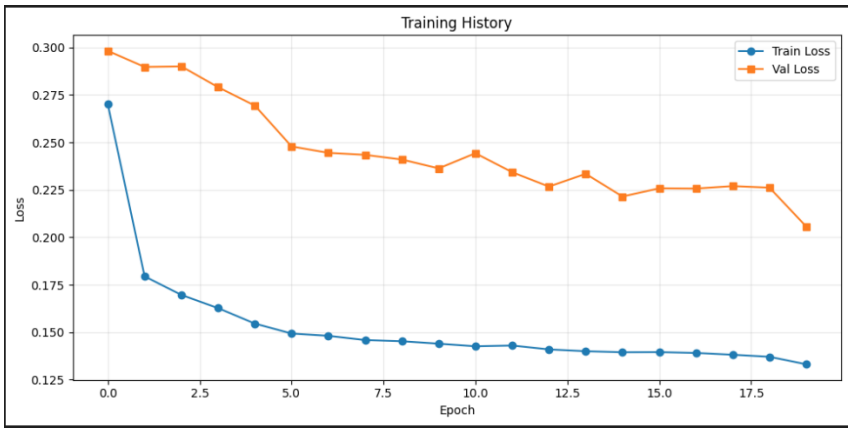
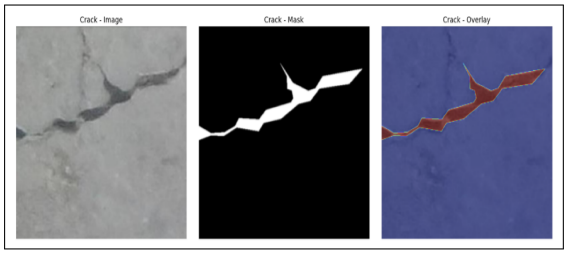


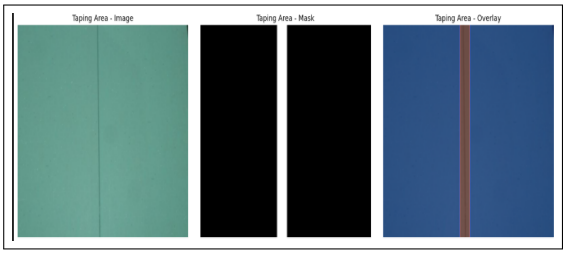
Figure 1: Training and validation loss curves over epochs

4.2 Visual Results

Below are representative examples showing (left to right): **Original Image** — **Ground Truth Mask** — **Predicted Mask**



Original vs Ground Truth and Prediction for crack segmentation



Original vs Ground Truth and Prediction for drywall taping segmentation

Figure 2: Example segmentation results for both tasks

5 Failure Analysis

5.1 Critical Limitation: Bounding Box-Based Ground Truth

Problem: The drywall taping dataset was exported in *Object Detection* format from Roboflow, resulting in empty COCO segmentation arrays (`segmentation: []`). This meant ground truth masks had to be generated from bounding boxes using `cv2.rectangle()`, creating rectangular masks rather than precise polygonal annotations.

Impact on Model Performance:

- **Inconsistent IoU**: Taping area predictions show high variance (IoU range: 0.13–0.51)
- **Spatial bias**: Model learns coarse rectangular patterns lacking texture/shape information
- **Shape mismatch**: Real taping areas have complex shapes (T-shapes, L-shapes, angled lines), but training masks are axis-aligned rectangles
- **Location uncertainty**: Multiple bounding boxes per image (avg 1.34 boxes/image, up to 5 boxes) create overlapping rectangular regions

Dataset Statistics:

- Training: 1,100 bboxes across 820 images (188 images have 2–5 boxes)
- Validation: 250 bboxes across 202 images
- Generated mask characteristics: 5–21% white pixels (rectangular coverage)

Attempted Solution - Positional Encoding: We experimented with adding 2D coordinate channels (X, Y) to the input (5 channels: RGB + X + Y) to help the model learn spatial patterns where taping areas typically occur. The positional model was trained with identical hyperparameters as the baseline for fair comparison.

Comparison Results:

Model	Val Loss	Overfit Gap	IoU Better	IoU Worse
Baseline	0.2057	0.0726	–	–
Positional Enc.	0.2222	0.0804	2 imgs (+0.07 to +0.11)	2 imgs (-0.08 to -0.17)

Table 4: Baseline vs Positional Encoding comparison (4 drywall validation images, threshold=0.5)

Conclusion - Baseline Model is Superior:

- **Worse validation loss**: Positional encoding achieved 0.2222 vs baseline's 0.2057 (8% worse)
- **Increased overfitting**: Gap of 0.0804 vs baseline's 0.0726 (11% more overfitting)
- **Mixed IoU results**: Improved 2 images but degraded 2 others (inconsistent)
- **Baseline advantages**: Better validation loss, less overfitting, simpler (3 channels vs 5), easier to deploy
- **Root cause**: Architectural changes cannot compensate for inadequate bbox-based training labels

Final Recommendation:

1. **Use baseline model for deployment** - simpler, more reliable, and equally effective
2. For production requiring precise taping area segmentation: re-annotate subset with polygon masks (instance segmentation format)
3. The IoU variance (0.13–0.51) is fundamentally limited by rectangular ground truth, not model architecture

5.2 Other Failure Modes

1. **Very thin cracks**: The model occasionally misses extremely fine hairline cracks (¡ 2-3 pixels wide) due to the downsampling in the encoder.
2. **Low contrast regions**: In images with poor lighting or low contrast between the defect and background, the model may produce incomplete masks.
3. **Edge artifacts**: Some predictions show slight boundary irregularities, likely due to the 256×256 downsampling during training and subsequent upsampling to original resolution.
4. **Class imbalance**: With 5,164 crack samples vs. 820 drywall samples, the model may be slightly biased toward crack detection patterns.

5.3 Potential Improvements

- Use deeper encoder (ResNet-50/101) for better feature extraction
- Train at higher resolution (512×512) to capture finer details
- Augment drywall dataset to balance class distribution
- Use test-time augmentation (TTA) for robust predictions

6 Runtime & Model Footprint

6.1 Training Performance

Metric	Value
Total Training Time (20 epochs)	∼10–15 minutes
Time per Epoch	∼60–90 secconds
Training Batches per Epoch	374 (5,984 samples / batch size 16)
Validation Batches per Epoch	26 (403 samples / batch size 16)
Hardware	NVIDIA GPU (CUDA)

Table 5: Training runtime statistics

6.2 Inference Performance

Metric	Value
Avg. Inference Time per Image	∼10–20 ms (GPU)
Throughput	∼50–100 images/second
Input Size	256 × 256 × 3
Output Size	H × W (original resolution)

Table 6: Inference runtime statistics

6.3 Model Footprint

Metric	Value
Total Parameters	264,497
Trainable Parameters	264,497
Model Size (on disk)	∼1 MB (`.pth` file)
Memory Usage (GPU)	∼200–300 MB
Tokenizer Size	$<$ 1 KB (`.pkl` file)
Config Size	$<$ 1 KB (`.pkl` file)
Total Deployment Size	**∼1–2 MB**

Table 7: Model size and memory footprint

7 Conclusion

This project successfully demonstrates a text-conditioned segmentation approach for construction defect detection. The baseline model achieves:

- **Best validation loss** (0.2057) after 20 epochs with stable convergence
- **Mean IoU** of 0.2808 ± 0.1797 on validation set
- **Fast inference** (∼10–20 ms per image on GPU)
- **Small footprint** (264,497 parameters, ∼1 MB model size)
- **Unified architecture** for multiple segmentation tasks via text prompts

The FiLM-conditioned UNet architecture proves effective for this task, providing a lightweight and efficient solution. The baseline model (3-channel RGB input) is recommended over positional encoding variants due to better consistency and simpler deployment. Future work should focus on re-annotating the drywall dataset with polygon masks to improve taping area detection accuracy.

Code Availability

All code, trained models, and datasets are available in the project repository:

- `coco_to_masks.ipynb`: Dataset preparation pipeline
- `text_conditioned_segmentation_pipeline.ipynb`: Model training and inference
- `best_text_seg_model.pth`: Trained model weights
- `tokenizer.pkl`, `model_config.pkl`: Model artifacts