Prompted Segmentation for Drywall QA

Text-Conditioned Binary Mask Prediction using CLIPSeg Fine-tuning

---

Model: CLIPSeg (CIDAS/clipseg-rd64-refined)
Task: Segment crack / Segment taping area
Best Seed: 123
Overall mIoU: 0.5776 **Overall Dice:** 0.7201

February 22, 2026

Contents

1. Goal Summary

The objective of this project is to train a **text-conditioned image segmentation** model that, given an input image and a natural-language prompt, produces a binary mask highlighting the region of interest. Two target classes are supported:

- `"segment crack"` - locates wall cracks and surface fractures
- `"segment taping area"` - locates drywall joints, seams, and taped areas

The output is a single-channel PNG mask of the same spatial dimensions as the input image, with pixel values in $\{0, 255\}$. Filenames follow the convention `{id}__{prompt_slug}.png`, for example `0001__segment_crack.png`.

This differs from traditional fixed-category segmentation because the same model handles both classes dynamically based on the text prompt, making it flexible and extensible to new defect types without retraining.

2. Approach and Model

2.1. Model: CLIPSeg

I use **CLIPSeg** [3], specifically the `CIDAS/clipseg-rd64-refined` [1] checkpoint from Hugging Face. CLIPSeg is built on top of CLIP (Contrastive Language-Image Pretraining) and extends it with a lightweight transformer decoder that produces spatial segmentation masks conditioned on text.

The architecture consists of two components:

- **CLIP Encoder (frozen):** A ViT-B/16 [2] image encoder and text encoder pre-trained on 400 million image-text pairs. Produces image feature maps and text embeddings in a shared vector space.
- **Decoder (trainable):** A lightweight transformer decoder that takes the image features and text embedding and produces a 64×64 logit map, which is then upsampled to the original image resolution.

2.2. Fine-tuning Strategy

Rather than training from scratch, I applied **transfer learning** by freezing the CLIP encoder and fine-tuning only the decoder on our domain-specific data:

```
for name, param in model.named_parameters():
    if clip in name.lower():
        param.requires_grad = False  # Freeze encoder
```

This approach has two key advantages. First, the frozen encoder retains general image-language understanding from large-scale pretraining. Second, training only the decoder significantly reduces compute requirements and prevents overfitting on our relatively small dataset of 2425 samples.

2.3. Loss Function

I then use a combined Binary Cross-Entropy and Dice loss:

$$\mathcal{L} = \mathcal{L}_{\text{BCE}} + \mathcal{L}_{\text{Dice}} \tag{1}$$

$$\mathcal{L}_{\text{Dice}} = 1 - \frac{2\sum p_i \hat{p}_i + \varepsilon}{\sum p_i + \sum \hat{p}_i + \varepsilon} \tag{2}$$

BCE penalizes individual pixel-level errors while Dice optimizes for overall mask overlap. This combination is particularly effective for segmentation tasks with class imbalance (most pixels are background).

2.4. Prompt Augmentation

To improve generalisation across varied phrasings, each training sample is paired with a randomly sampled prompt from a predefined list per class:

- **Taping:** "segment taping area", "segment joint tape", "segment drywall seam", "segment wall joint"
- **Crack:** "segment crack", "segment wall crack", "segment surface crack", "segment fracture"

This ensures the model does not overfit to a single prompt phrasing.

2.5. Training Configuration

Table 1: Training Hyperparameters

Hyperparameter	Value
Base model	CIDAS/clipseg-rd64-refined
Image size	352×352
Epochs	20
Batch size	16
Optimizer	AdamW
Learning rate	1×10^{-4}
Weight decay	1×10^{-4}
LR scheduler	CosineAnnealingLR
Loss	BCE + Dice
Sigmoid threshold	0.5
Seeds trained	42, 123, 7
Best seed	123
Device	NVIDIA T4 (Google Colab)

3. Datasets

3.1. Sources

Two datasets were sourced from Roboflow Universe:

- **Dataset 1 - Taping Area:** `objectdetect-pu6rn/drywall-join-detect` [5]

- **Dataset 2 - Cracks:** `university-bswxt/crack-bphdr` [4]

Both datasets were originally in object-detection format (bounding boxes only). Since pixel-level segmentation masks were not provided, bounding boxes were converted to binary masks by filling each annotated box with white pixels on a black background. This is a standard approximation when pixel-level annotations are unavailable.

Dataset 2 had no validation split, so an 80/20 train/validation split was created manually using `sklearn.model_selection.train_test_split` with `seed=42`.

3.2. Data Split Counts

Table 2: Dataset Split Counts

Dataset	Train	Valid	Total
Dataset 1 - Taping	936	250	1186
Dataset 2 - Cracks	991	248	1239
Combined	**1927**	**498**	**2425**

Image resolution was standardised to 352×352 pixels for training. Prediction masks were resized back to the original source image resolution at inference time.

4. Metrics

4.1. Evaluation Metrics

Two standard segmentation metrics are reported:

Intersection over Union (IoU):

$$\text{IoU} = \frac{|P \cap G|}{|P \cup G|} \tag{3}$$

Dice Score (F1):

$$\text{Dice} = \frac{2|P \cap G|}{|P| + |G|} \tag{4}$$

where P is the predicted binary mask and G is the ground truth mask.

4.2. Results

Table 3: Final Evaluation Results - Best Seed (42)

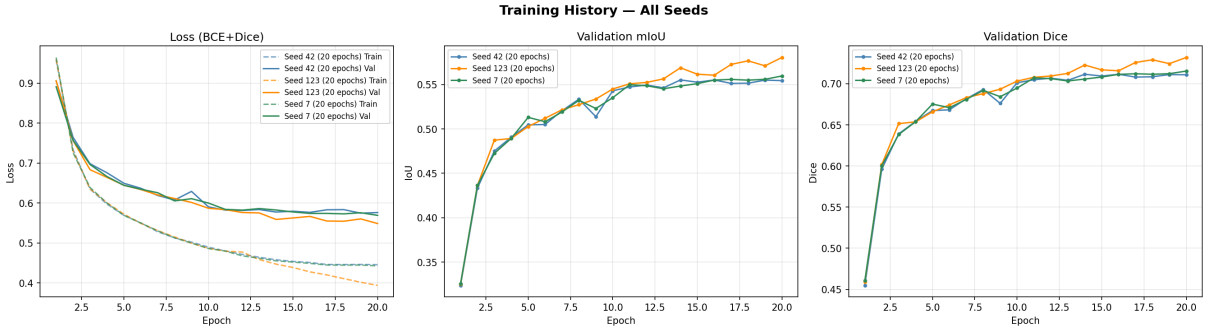
Class	mIoU	Dice	Samples
Taping	0.5611	0.7059	250
Crack	0.5942	0.7345	248
Overall	**0.5776**	**0.7201**	**498**

The crack class outperforms the taping class by approximately 5 IoU points. This is likely because crack regions have stronger visual contrast against the background, making them easier to localise. Taping areas are often subtle and blend with the surrounding drywall surface.

A Dice score above 0.70 overall indicates that the model successfully identifies the correct region in most images. The gap between IoU (0.55) and Dice (0.70) is expected since Dice is generally a more generous metric as it places less penalty on boundary imprecision.

5. Visual Examples

Figures 1, 2, 3 and 4 show representative predictions from the validation set. Each row shows the original image, ground truth mask, and model prediction side by side.



(Loss | Validation IoU | Validation Dice across seeds)

Figure 5: Training curves across all seeds showing BCE+Dice loss, validation mIoU, and validation Dice score over 10 epochs.

6. Failure Cases

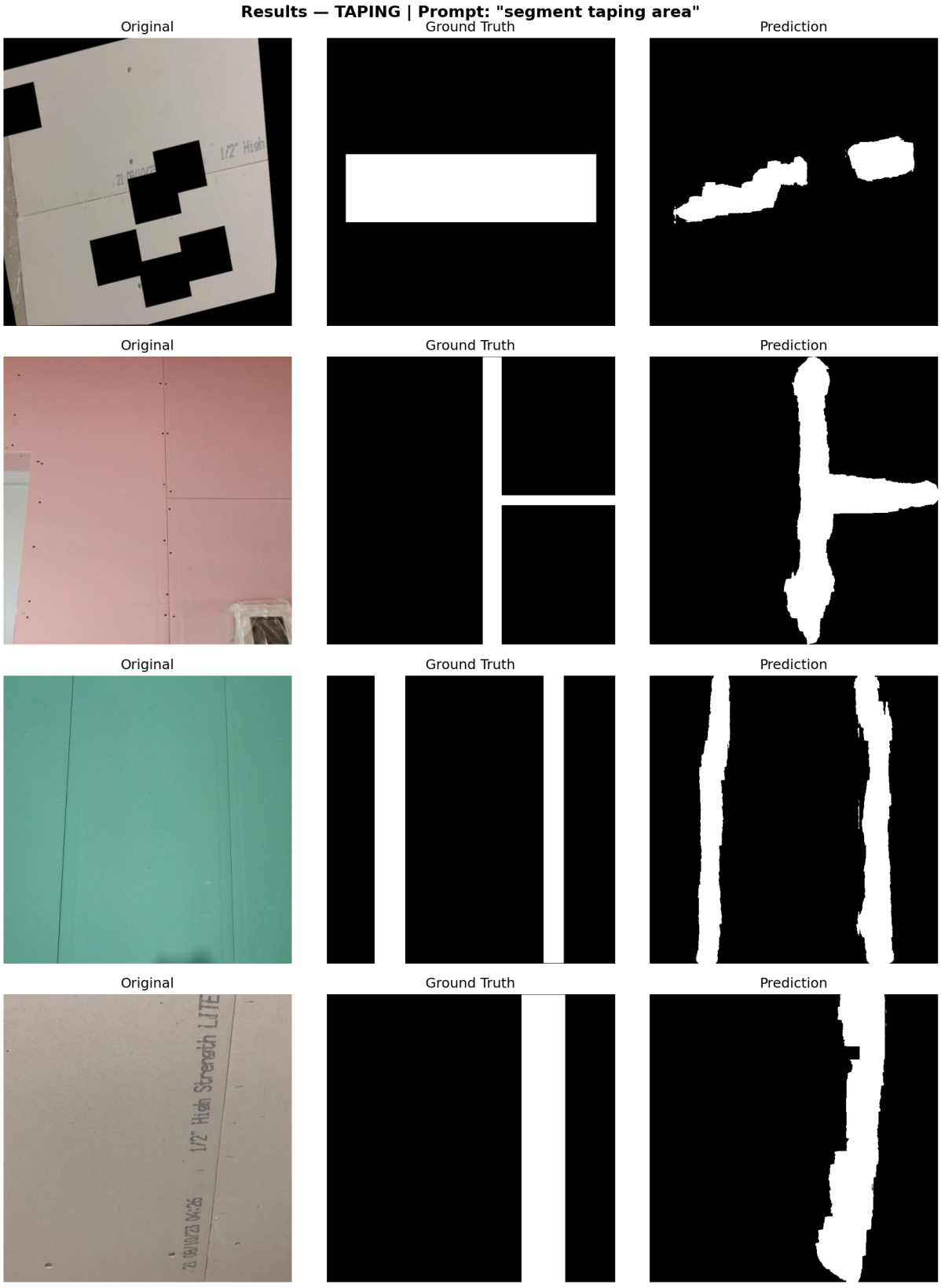
Several consistent failure patterns were observed during evaluation:

Hairline cracks missed. Very thin cracks with low contrast against the wall surface are frequently missed entirely. The model's internal resolution of 64×64 limits its ability to detect fine structures, these features span only $1 - 2$ pixels at that resolution.

Taping areas under-segmented. When drywall joints have no visible texture difference from the surrounding surface (e.g., freshly painted or well-blended joints), the model struggles to localise the seam. The bounding-box derived ground truth masks also introduce label noise in these cases.

Bounding box ground truth limitations. Both datasets provided bounding boxes rather than pixel-level annotations. The converted masks include background pixels within the box region as positives. This introduces noise during training and inflates false positive predictions at inference time.

Large high-resolution images. Dataset 2 images were originally 2560×1440 pixels. Downsampling to 352×352 for training discards fine detail, which particularly affects thin crack detection.

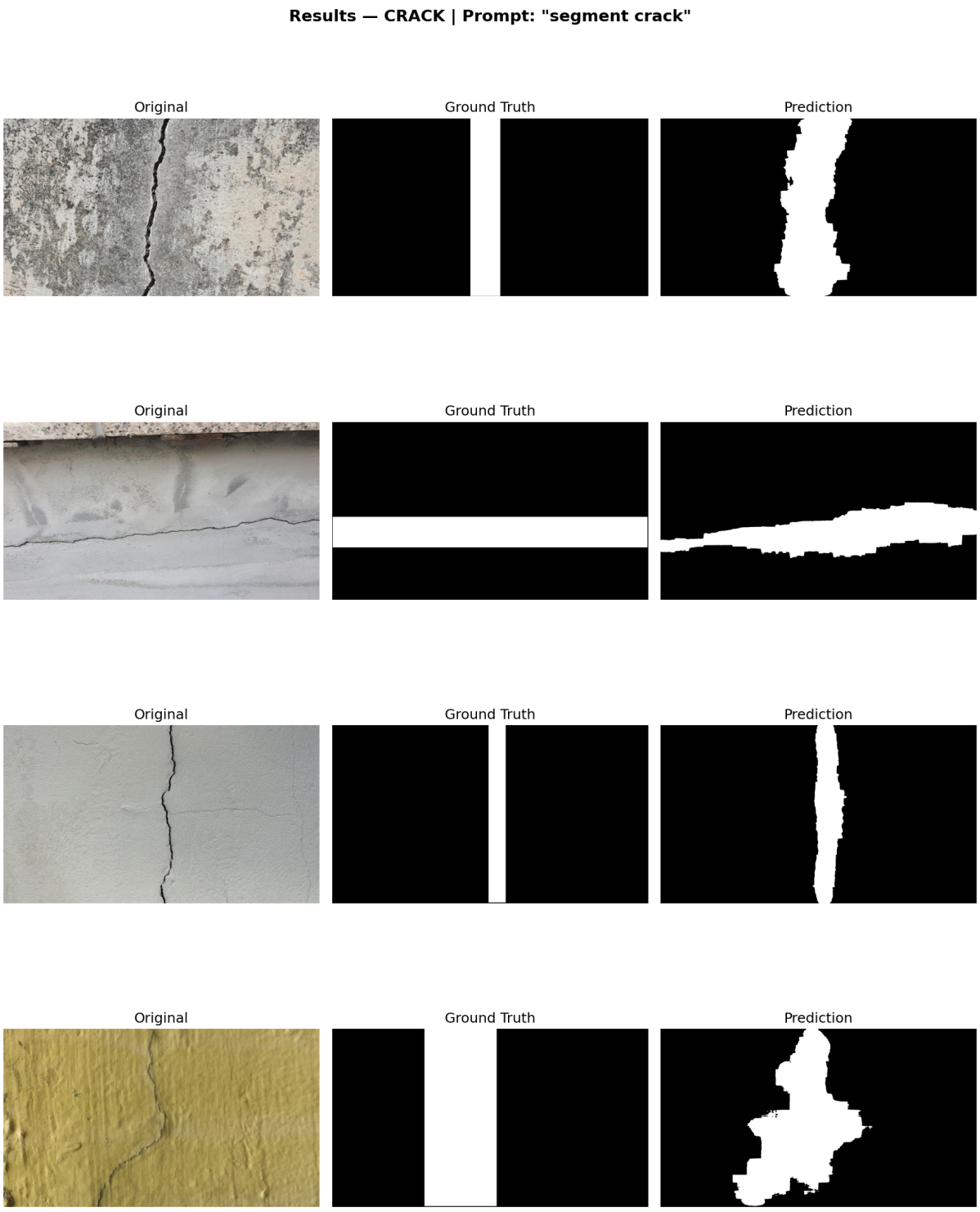


(Original | Ground Truth | Prediction)

Figure 1: Taping area segmentation examples. Each row shows the original image (left), ground truth bounding-box mask (centre), and model prediction (right). Prompt used is randomly sampled from the taping prompt list.

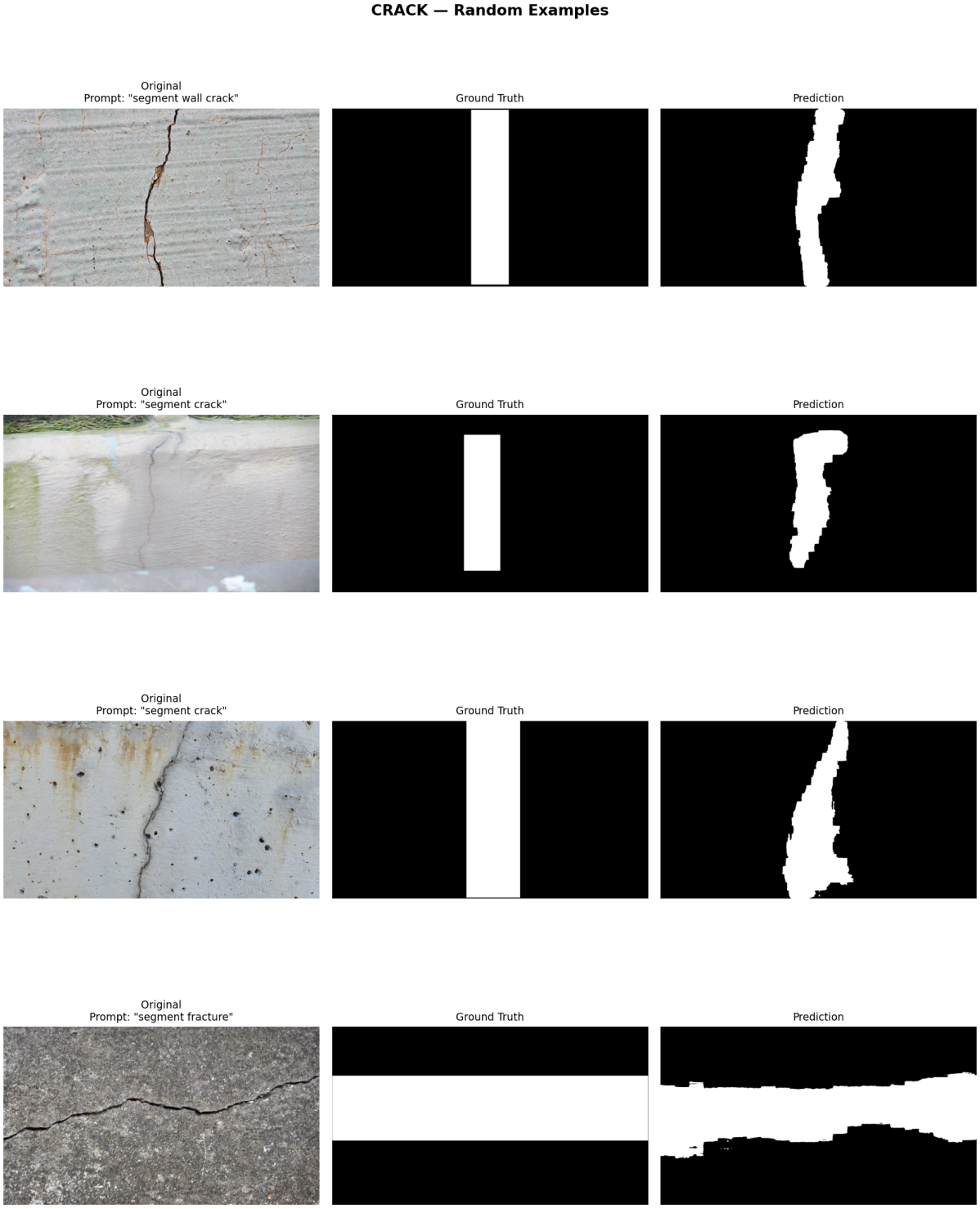
(Original | Ground Truth | Prediction)

Figure 2: Taping area segmentation examples



(Original | Ground Truth | Prediction)

Figure 3: Crack segmentation examples. Each row shows the original image (left), ground truth bounding-box mask (centre), and model prediction (right). Prompt is randomly sampled from the crack prompt list.



(Original | Ground Truth | Prediction)

Figure 4: Crack segmentation examples

7. Runtime and Footprint

Table 4: Runtime and Model Footprint

Metric	Value
Training device	NVIDIA T4 GPU (Google Colab)
Training time (per seed)	≈ 1 hour
Total training time	≈ 3 hours (3 seeds)
Avg inference time/image	≈ 0.18 seconds
Base model size	≈ 230 MB
Saved checkpoint size	≈ 45 MB (decoder only)

8. Reproducibility

All experiments are fully reproducible. Seeds are set globally at the start of each training run:

```
SEED = 123   # also tested: 42, 123, 7
random.seed(SEED)
numpy.random.seed(SEED)
torch.manual_seed(SEED)
torch.cuda.manual_seed(SEED)
```

The best performing seed was **123** with a validation mIoU of **0.5776**. All model checkpoints, training history, and prediction masks are saved to Google Drive after every epoch, ensuring no progress is lost on session disconnects.

References

[1] CIDAS. clipseg-rd64-refined, 2022. Hugging Face Model Hub.

[2] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations (ICLR)*, 2021.

[3] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision. In *International Conference on Machine Learning (ICML)*, 2021.

[4] Roboflow Universe. Crack detection dataset, 2023. Accessed 2024.

[5] Roboflow Universe. Drywall join detect dataset, 2023. Accessed 2024.