Barmak Lab Research Report

Ming Gong (mg4264)

Department of Applied Physics and Applied Mathematics, Columbia University

Spring 2025

Abstract

Accurate tracing of grain boundaries in microscopy images is vital in material science, yet current models need more data and a more accurate loss function. In this report, we present a twofold contribution to improving grain-tracing U-nets. First, we introduce a systematic data augmentation pipeline that uses GIMP to crop each image and its corresponding ground-truth tracing into aligned sub-images. This approach expands the training set while preserving trace integrity. Second, we examine binary cross-entropy (BCE) loss and demonstrate its tendency to double-penalize slight misalignments. To address that, we develop two evaluation metrics: a binary "top-hat" criterion that rewards traces within a fixed-pixel dilation and a continuous normalized Gaussian dilation loss that smoothly interpolates reward and penalty based on distance. By constructing dilated reward masks and computing pointwise products within network outputs, we obtain loss curves that decrease monotonically over 0-300 training epochs, consistent with the trend of qualitative visual improvements. The evaluation function allows turnable tolerance through the Gaussian σ parameter. Together, these methods offer a robust framework for data generation and performance evaluation in U-net tasks.

Introduction

We use Unet models to predict grain boundaries. The current model predictions are very accurate, but there are a few areas where it can be improved. This report will have two parts:

1. Data Augmentation Pipeline: a systematic way to generate new training data from the existing data, to boost training data quantity and encourage more definitive predictions.
2. Custom evaluation metric: We analyze the current BCE loss functions and build an alternative from scratch, incrementally adding features to it. This gives us an objective, tunable way to compare models trained with and without the augmented data.

1. Data Augmentation Pipeline

The first task is to generate more training data from existing data. We extend our training set by splitting each original input image—and its corresponding ground-truth tracing—into smaller sub-images using GIMP.

1.1 Cropping the Image

The tracing only covers complete grains. Grains extending to the input's boundary are not traced in the ground truth tracing image. For each sub-image, we ensure that each sub-tracing contains as few boundaries as possible. However, including a few untraced areas is still fine, as the original model is trained with normal input but truth with cut boundaries.

A sub-image will typically include more than 5 grains. Some sub-images overlap in the center in order to capture the boundaries.

For example, in Figure 1(a), a tracing image may be cut into the following five sub-images.

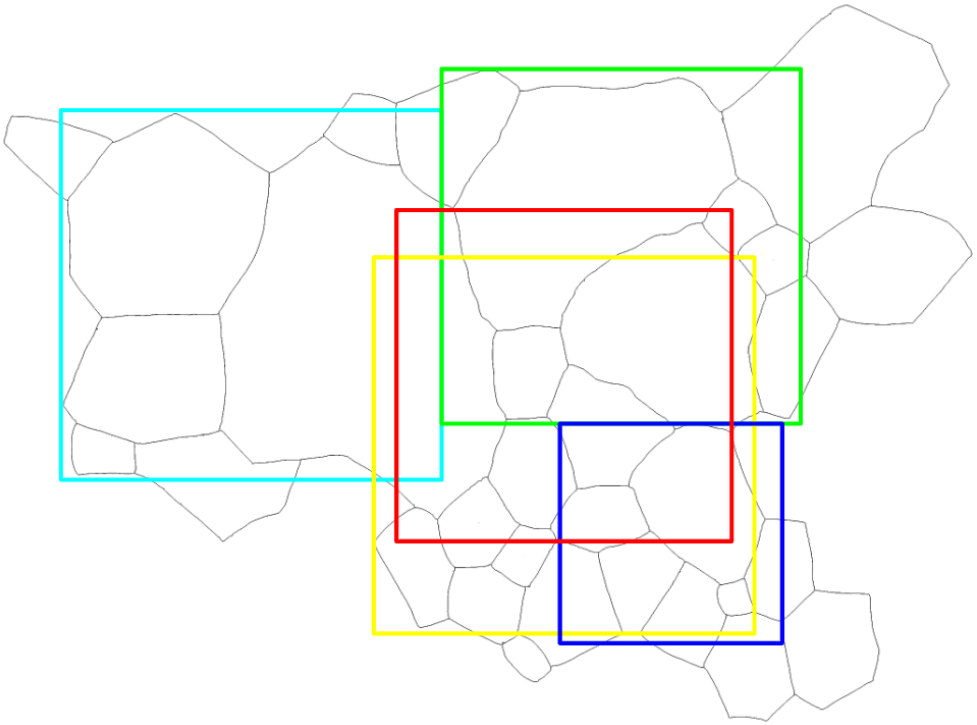


Figure 1(a). Splitting an image into different sub-images. Each colored box represents a sub-image.

Using GIMP, we applied the same crop on the original input images at different fields of view to ensure that the resulting images and tracings were still aligned.

1.2 Image Processing

The original cropped labels directly exported from GIMP have issues: The tracing lines are blurry. The images need to go through the following procedure shown in Figure 1(b):

1. Binarize the tracing
2. Dilate the tracing using a Gaussian kernel to 3 pixels

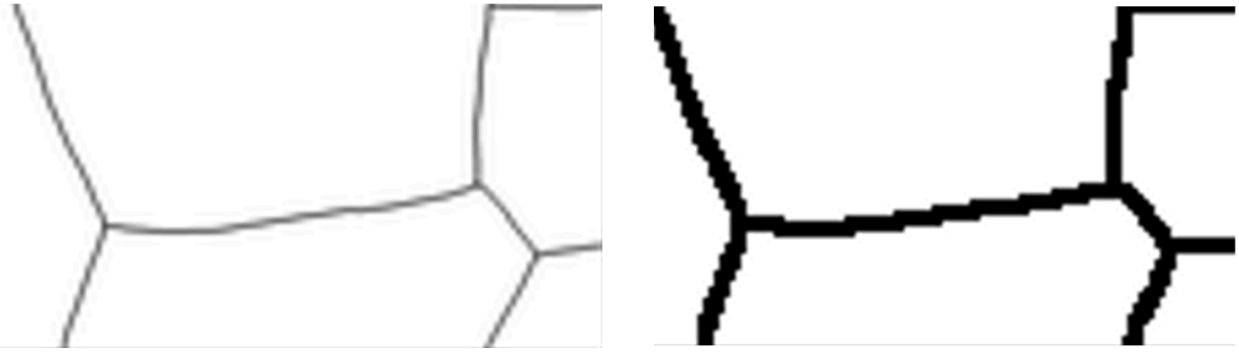


Figure 1(b). Local detail of a cropped tracing. The left is directly exported from GIMP, and the right is binarized and dilated to 3 pixels.

The dilation thickness can vary, and a future work will be to compare model training results under different truth tracing thicknesses.

1.3 Folder Structure and Naming Convention

Originally, images from different fields of view all pointed to the same tracing file. We fixed this by duplicating and renaming each tracing so that every image now has its own unique, one-to-one mapping to a corresponding tracing, shown in Figure 2.

```
Unset
training_validation_data/
└── validation/
    ├── xxxx_validation/
    |   └── aligned/
    |       ├── xxxx_n.png
    |       ├── xxxx_trace.png
    |       ├── image/   # cropped input images
    |       |   ├── xxxx_n_m.png
    |       └── label/   # matching cropped tracings
    |           ├── xxxx_n_m.png
    └── …
```

- Original input: `xxxx_n.png`
- Original tracing: `xxxx_trace.png`
- Cropped input: `image/xxxx_n_m.png` (where *m* is the cropping index)
- Cropped label: `label/xxxx_n_m.png`

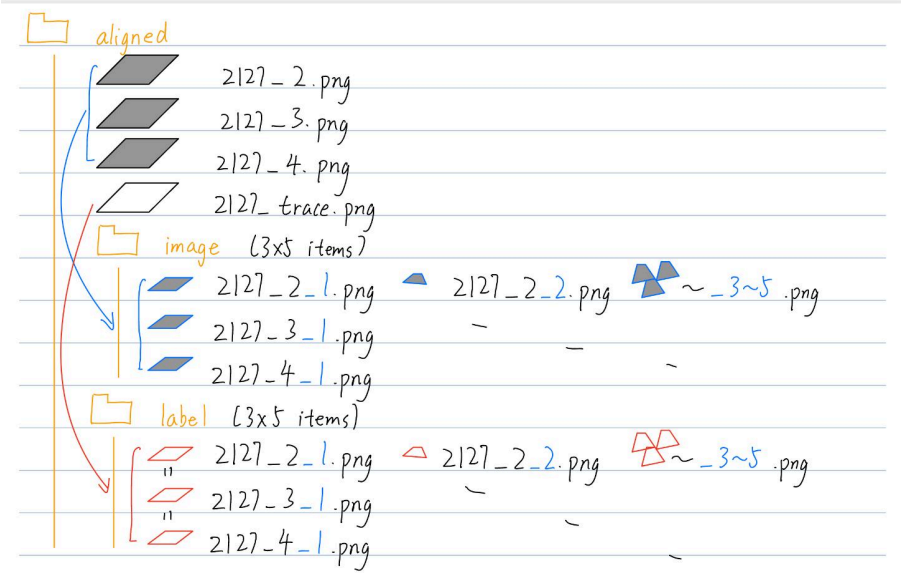


Figure 2. File folder organization of the cropped images and tracings.

MP: Make a note that the original 299 patches were prepared in the same way, and this is to extend the dataset

1.4 Result

We combined the cropped new data with the original ("OG") training set to create the **Gong_sum** dataset and likewise merged it with the Grae retracing set to form **GG_sum**.

All three datasets were used to train and infer at a fixed resolution of 1024×1024 over 90 epochs. The resulting predictions are presented in Figure 3.

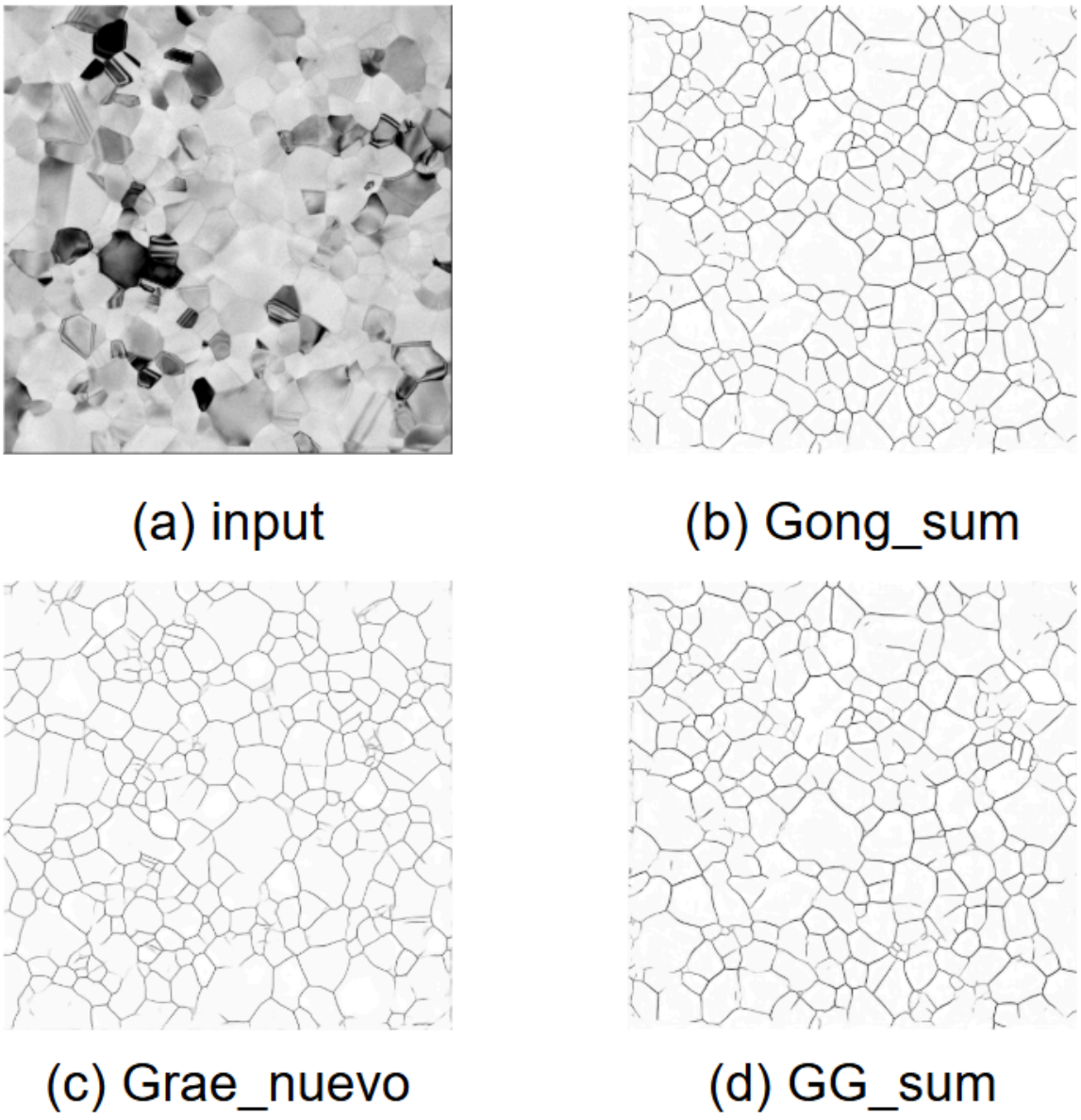


Figure 3. (a) input and (b-d) inference results using (b) OG, (c) Grae_nuevo, and (d) GG_sum training sets.

As in Figure 3(b and d), the augmented datasets (Gong_sum and GG_sum) produce more definite predictions with higher-contrast tracings. Neither model, however, fully captures the finest structural details.

2. Loss Function Evaluation

The original goal was to monitor how the model prediction quality evolves over training by evaluating a single input image against multiple model checkpoints (0-300), eventually developing a pipeline for multiple input images.

However, we encountered issues with the standard binary cross-entropy loss. In this section, we

1. Diagnose the problem of point-wise metrics like BCE for grain-tracing tasks
2. Design a custom loss function from scratch, from a simple top-hat dilation to adding Gaussian smoothing, adding complexity until we arrive at a metric that aligns with human judgment.

2.1 Workflow

Below is the workflow for preprocessing the input and tracing images to prepare data for the model and the evaluation function:

Input Preprocessing

1. Remove the image borders (scale bar, etc)
2. Resize the image to the output resolution

Tracing Preprocessing

1. Remove borders
2. Resize to the output resolution
3. Binarize to white tracing on a black background
4. Dilate the tracing lines to 3 pixels

No additional postprocessing, such as binary thresholding, was applied to the model output.

Evaluation

1. Inference
    - Load a preprocessed input image at the target resolution.
    - Run the model (for a given checkpoint/epoch) to produce the output mask.
2. Loss Evaluation
    - Apply the loss function to the output and ground-truth tracing.
    - Record this single “loss score” as that model’s performance on the image.

3. Epoch Sweep
    - Repeat steps 1–2 for each saved model epoch
    - Plot loss versus epoch to visualize training convergence.

2.2 Point-wise Loss Functions

We have been evaluating the predictions with BCE Loss, the loss function used for training.

BCE

BCE stands for binary cross-entropy, commonly used in binary classification problems. For each pixel, it measures the difference between predicted probabilities and the actual binary labels. The final BCE score is simply the mean over all N pixels.

$$BCE = -\frac{1}{N}\sum_{i=0}^{N} y_i \cdot log(\hat{y}_i) + (1 - y_i) \cdot log(1 - \hat{y}_i)$$

where y is the predicted output and $\hat{y}$ is the ground truth

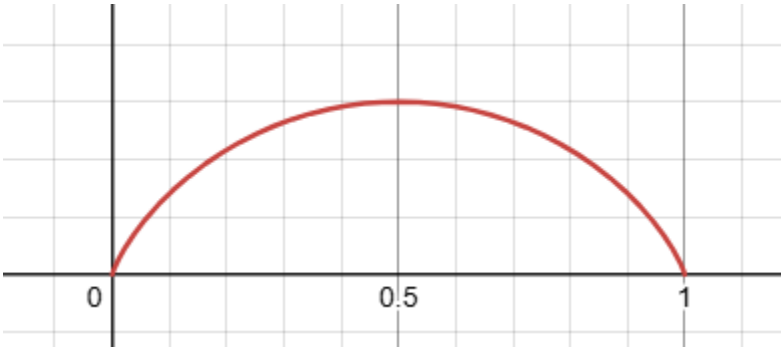


Figure 4: BCE loss of a single pixel against itself. The horizontal axis is the pixel brightness.

From Figure 4, BCE Loss may not be zero even if both images match exactly. BCE rewards confident outputs (0 or 1) and penalizes anything in between. Concretely, when both pixels are 1, -log(1) = 0; when both pixels are 0, log(1-0) = 0. However, neither log term will evaluate to zero for pixel values in between (gray), so the loss remains positive.

2.3 Issue with BCE Loss

The prediction was evaluated against the ground truth with `nn.BCELoss()` as the loss function. As in Figure 5(a), over increasing different epochs, the loss diverged.

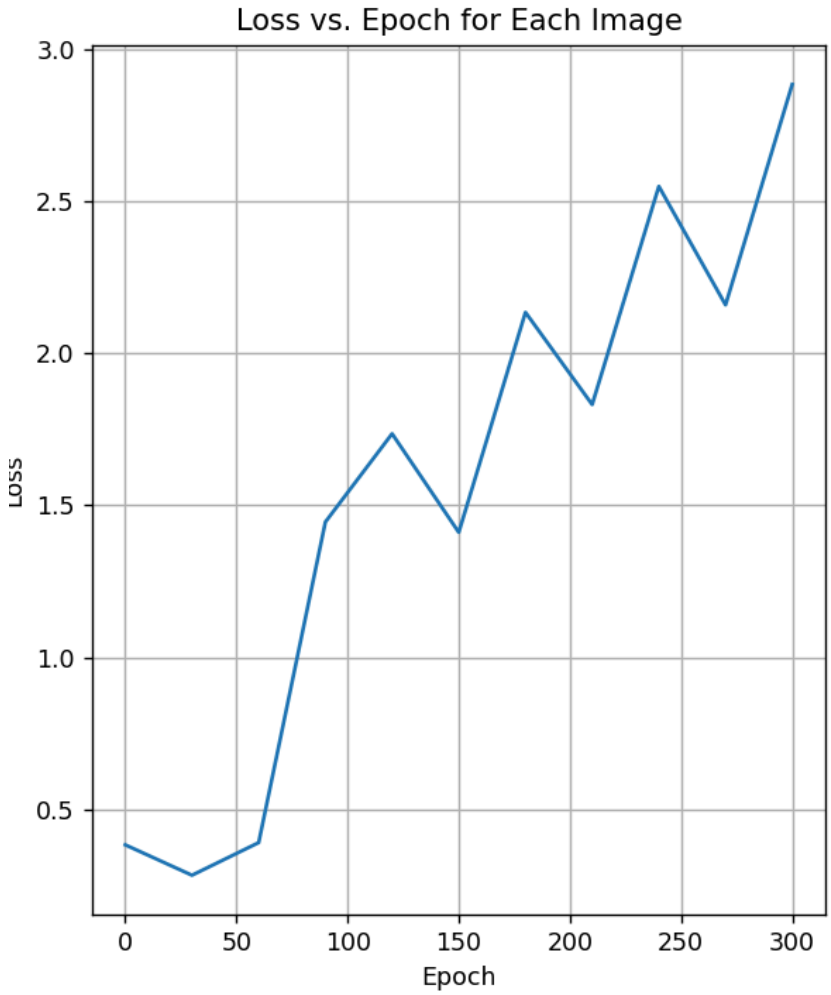


Figure 5(a). BCE Loss vs Epoch, with ground truth tracing = 3 pixels and output resolution 1024*1024

To isolate the problem, in Figure 5(b), we showed the model outputs at different epochs of the same input image at 1024 output resolution.

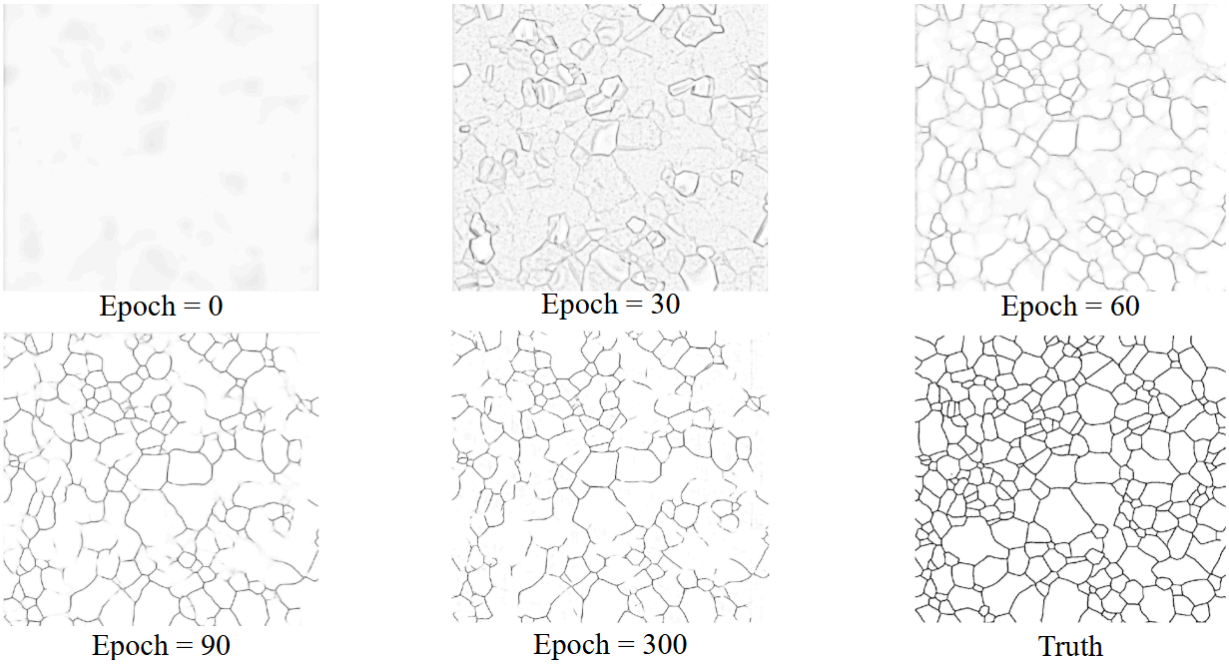


Figure 5(b): Model inference of the same input image, compared to the ground truth (lower right)

In the top row, the 0-epoch model produces a very blurry output. By 30 epochs, it correctly traces the overall grain shapes—though its coloring is still rough. From 60 to 90 epochs, the network refines finer structural details and yields much sharper boundaries. After 90 epochs, improvements are limited to minor, local adjustments: by 300 epochs, the predictions closely match the ground truth, aside from some very fine tracings that remain incomplete. Overall, the outputs steadily converge toward the reference masks as training progresses.

However, the BCE loss function produced counterintuitive results. To find out why, we compared and output and traced images pixel-wise to find out the origin of BCE loss.

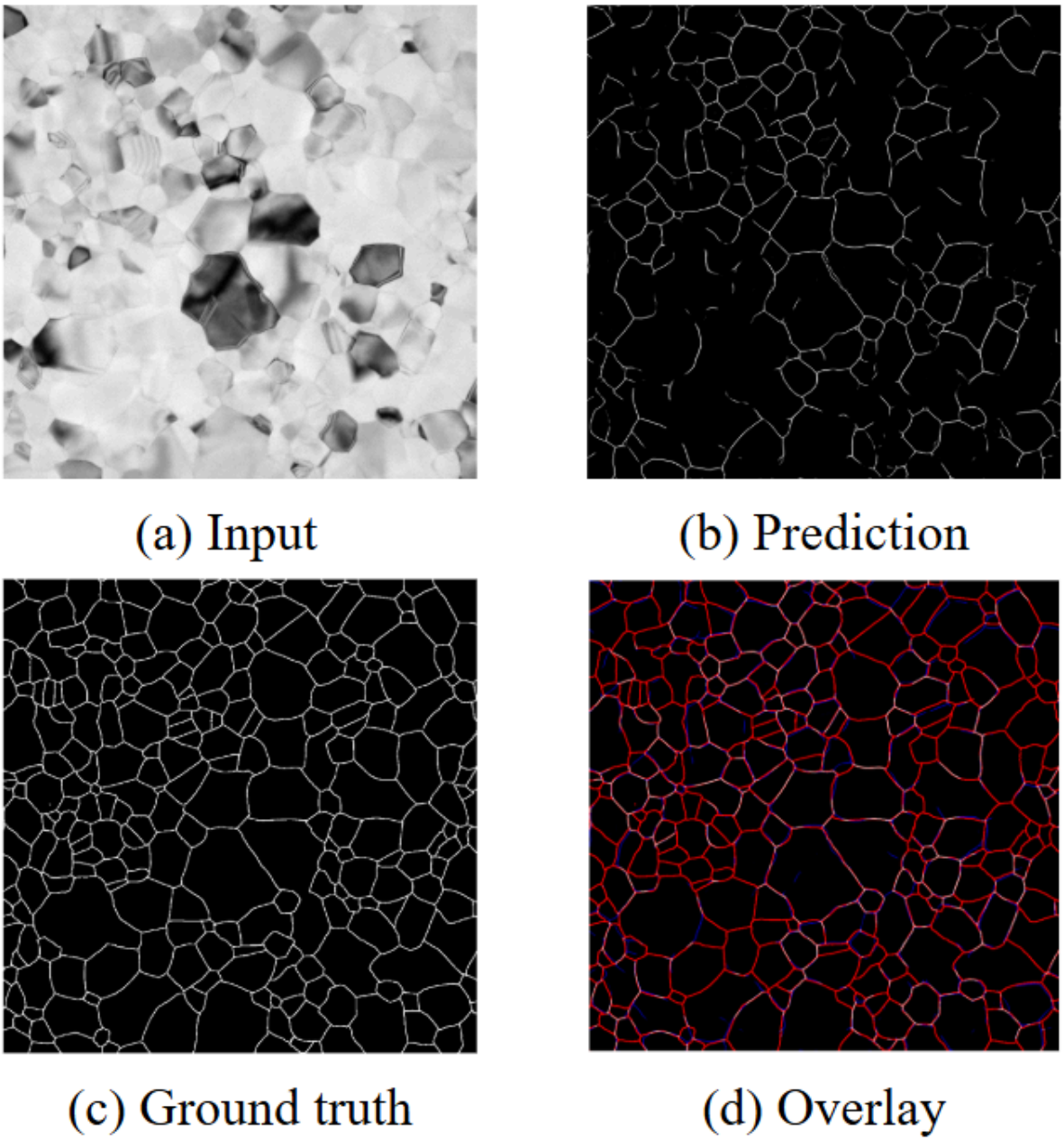


Figure 6: (a) input, (b) prediction, (c) ground truth, and (d) overlay of prediction on ground truth. Results are from inference using a model trained for 300 epochs with an output resolution of 1024.

Figure 6 displays the input, output, and truth images side-by-side. They are overlaid in Figure 6(d), where each pixel represents the following:

- White: correct boundary prediction (true positive)
- Black: correct background prediction (true negative)
- Red: Missed prediction (false negative)
- Blue: Incorrect prediction (false positive)

The white and black pixels will have zero contribution to the loss, but both red and blue are penalized by BCE.

The overlaid image has a lot of red and blue segments, meaning it's heavily penalized by BCE.

In Figure 7, the predicted results are very close to the ground truth, with the same shape and topology, but most lines are off by a few pixels. Red and blue **double-penalize** the missed pixels. Only a small portion of the overlap (white) is not penalized.

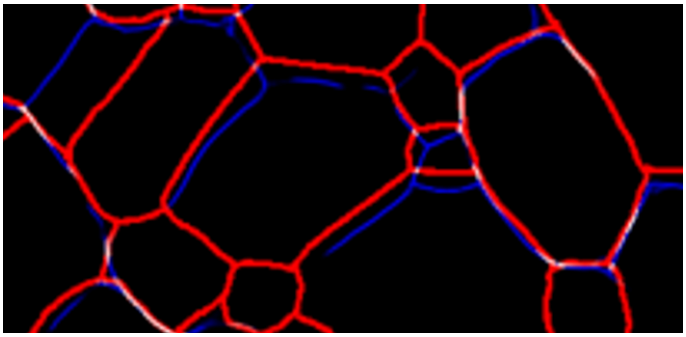


Figure 7: a local detail of Figure 6(d).

Also, when looking at the overlaid images globally, the red and blue lines cannot be told apart, but the pixel-wise BCE loss function will catch the slight difference and double-penalize it.

As a comparison, the inference at 50 epochs is evaluated below in Figure 8:

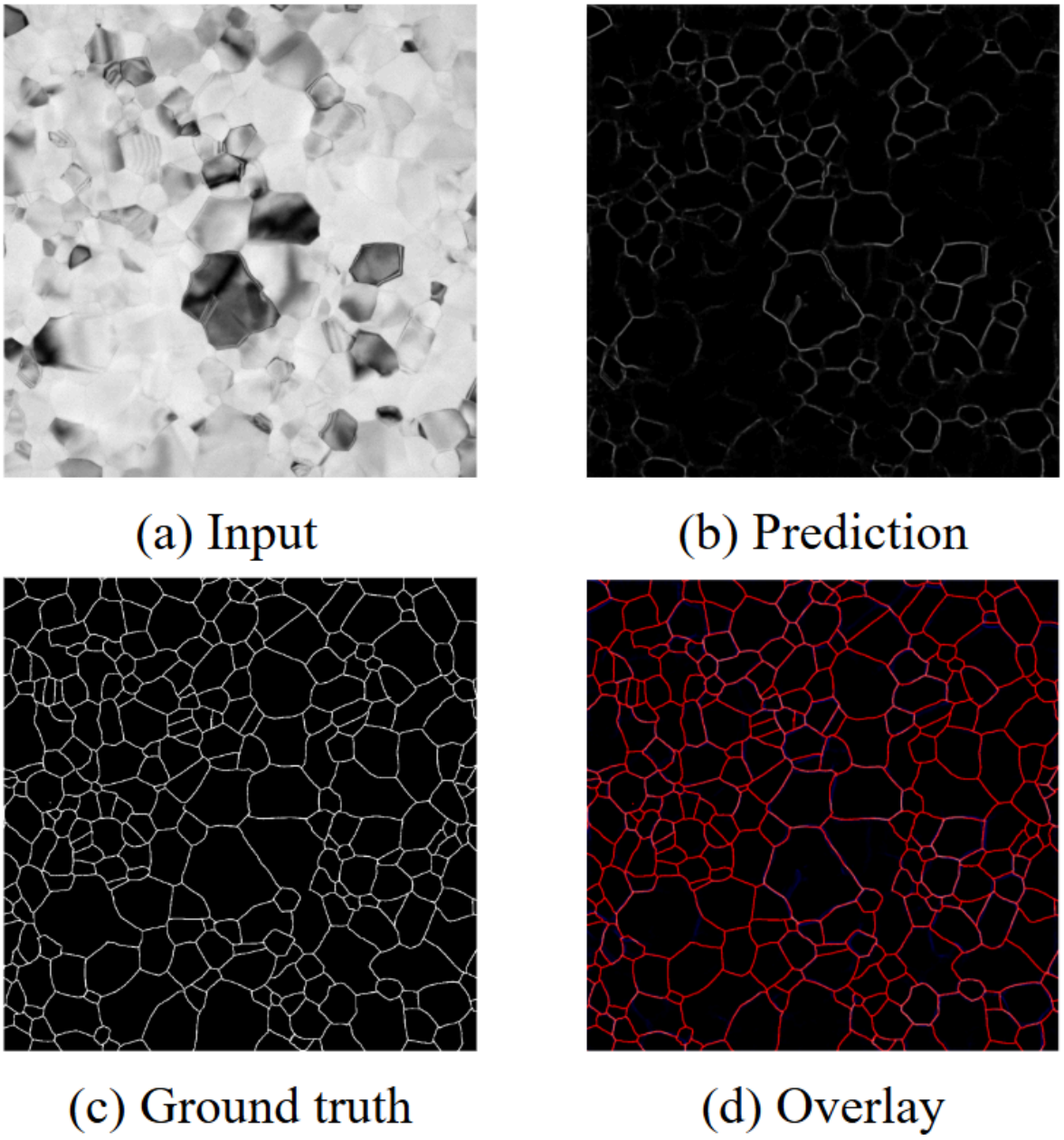


Figure 8: (a) input, (b) prediction, (c) ground truth, and (d) overlay of prediction on ground truth. Results are from inference using a model trained for 50 epochs with an output resolution of 1024.

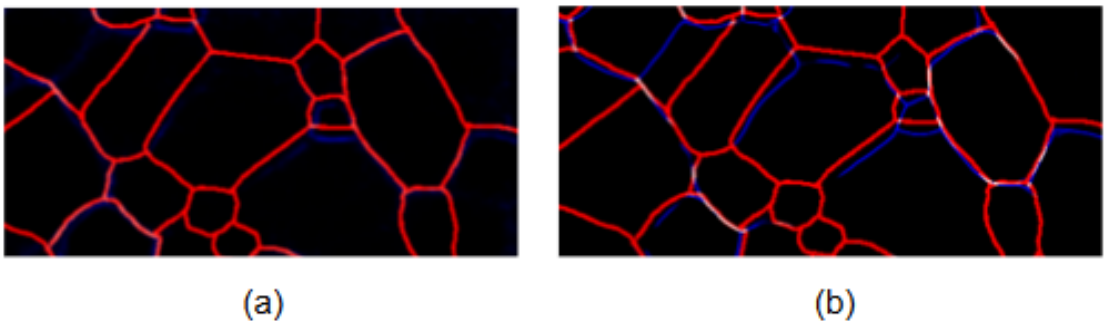


Figure 9: local details of (a) Figure 8(d) (50 epochs) and (b) Figure 6(d) (300 epochs)

Figure 9 compares the 50-epoch overlay and 300-epoch overlay side-by-side. At 50 epochs, the model's prediction is mostly ambiguous, with many disconnected dots. At 300 epochs, the prediction from certain lines resembles the actual grain boundaries.

However, for 50 epochs, since the model barely predicts anything, it is only penalized once for "missed prediction" (red), resulting in a BCE loss of 0.31.

For 300 epochs, since most predicted pixels miss the actual truth by a small margin, it is penalized both for "missed" (red) and "false positive" (blue), resulting in a higher BCE loss of 2.61!

Below, we isolated a few more problems, showing BCE loss is the actual root cause of the issue.

Truth thickness

In the section above, the overlaid images contain significantly more red pixels (false negatives) than blue pixels (false positives). Also, the tracing appears thicker (lines with more pixels) than the model prediction.

We tried to vary the thickness of the ground truth tracing. The truth tracing was dilated to 3 pixels and then skeletonized to 1 pixel (through repeated erosion to 1-px line widths). The magnitude of the loss dropped (due to fewer missed pixels), but the loss vs epoch still diverged in Figure 10.

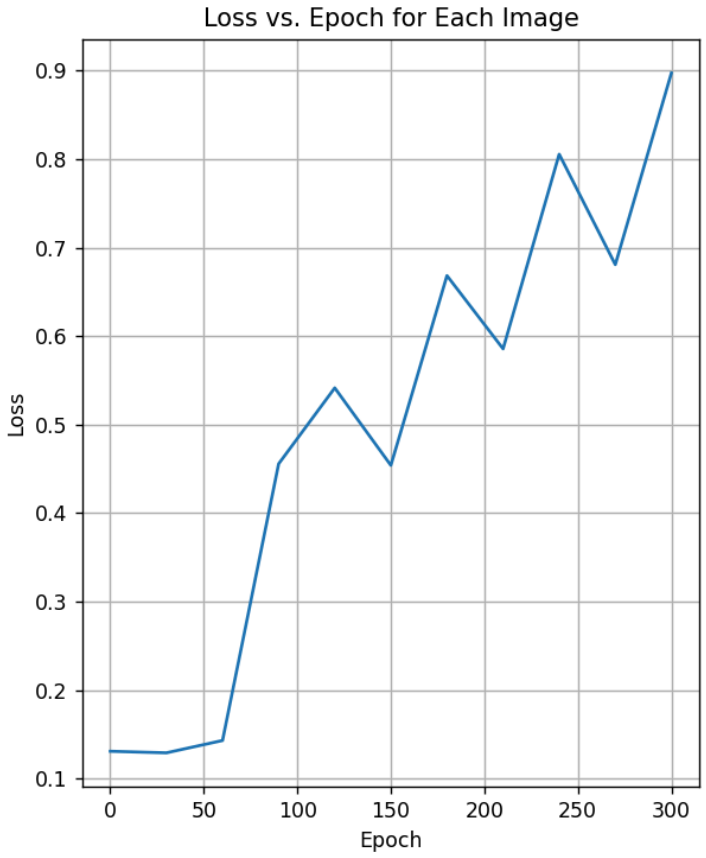


Figure 10. BCE Loss vs Epoch, with ground truth tracing = 1 pixel and output resolution 1024.

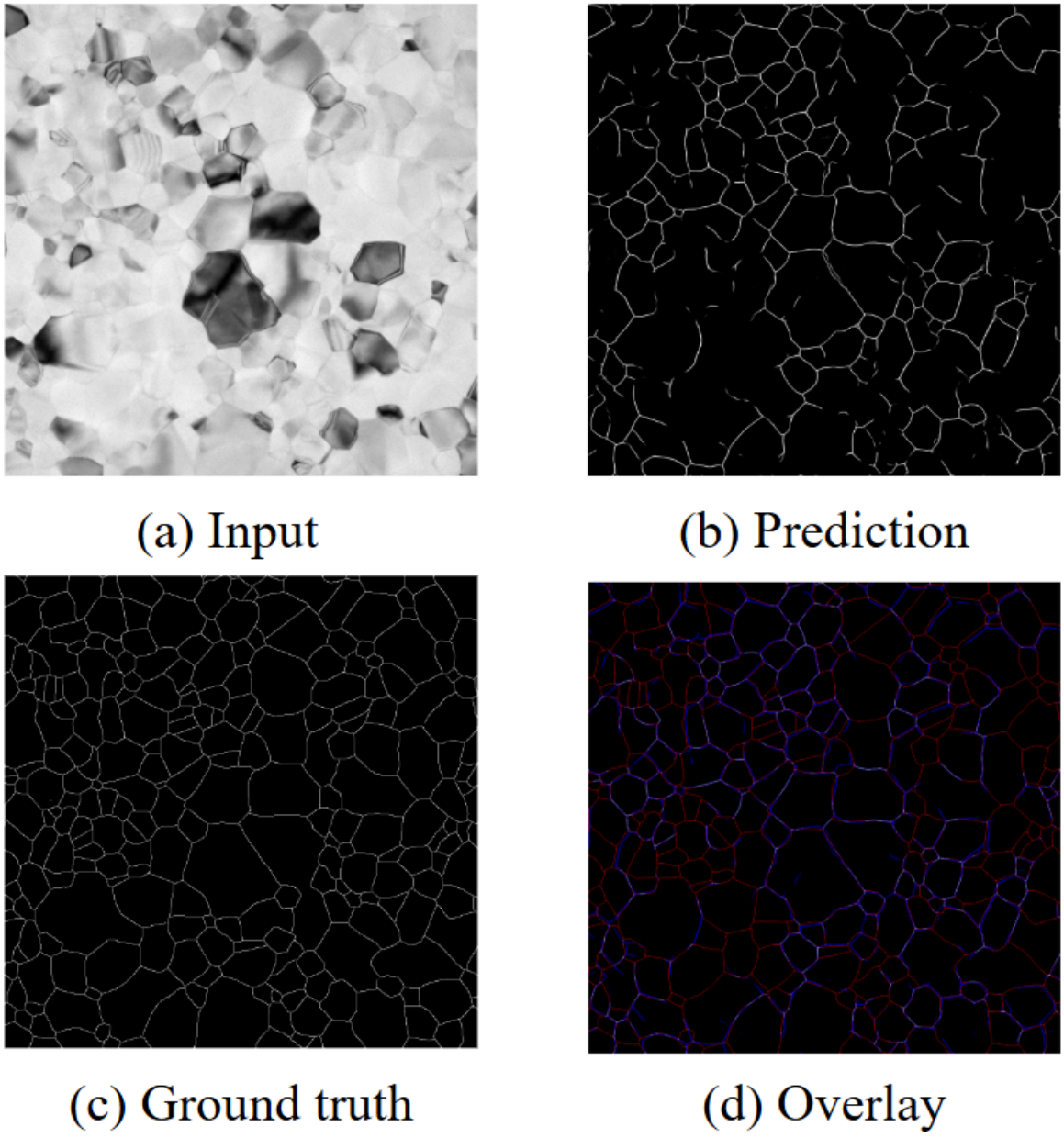


Figure 11: (a) input, (b) prediction, (c) 1-pixel-thick ground truth, and (d) overlay of prediction on ground truth. Results are from inference using a model trained for 300 epochs with an output resolution of 1024.

As in Figure 11, a thinner ground-truth tracing will reduce the number of false negatives, but it does not fundamentally solve the alignment problem. Each image merely has fewer misses. However, misalignments are still double-penalized. Since the tracing is thinner, misalignment is even more serious.

Output Resolution

The output resolution is also adjusted from 1024 to 256 to dilate each pixel and mitigate the misalignment issue. However, as in Figure 12, misalignment is still prevalent.

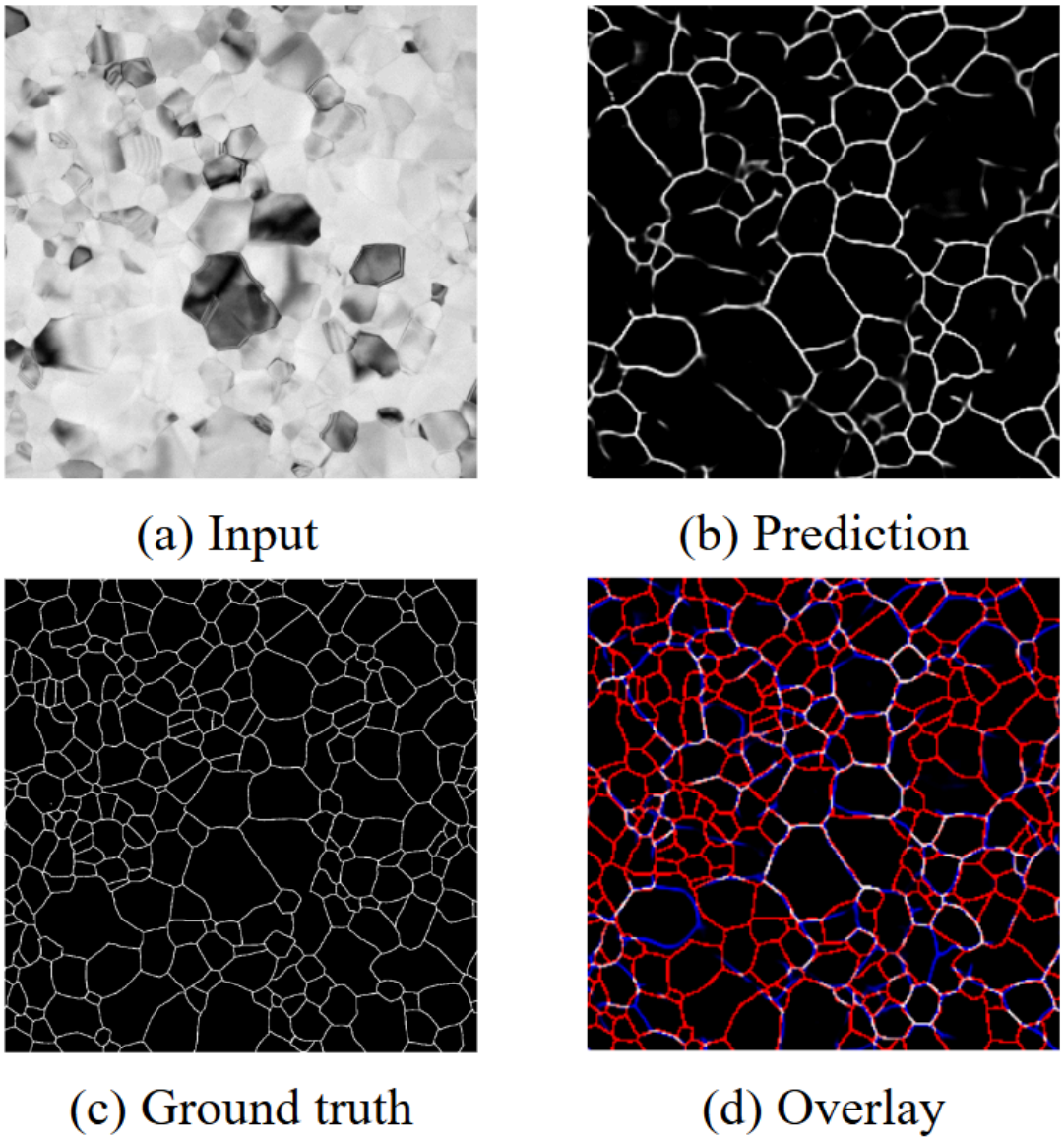


Figure 12: (a) input, (b) prediction, (c) ground truth, and (d) overlay of prediction on ground truth. Results are from inference using a model trained for 300 epochs with an output resolution of 256.

In summary, pixel-wise comparison methods, such as BCE, have the following problems:

- Even tiny misalignments (red vs. blue traces) count as full errors, doubling the loss
- Predicting a blank mask can yield deceptively low BCE.

3. Top-Hat Classification

This section introduces the **top-hat evaluation** as an alternative to BCE loss. Top-hat classification builds a spatial tolerance "halo" around the ground-truth tracing:

- **Predictions inside** the dilated mask are **rewarded** (green).
- **Predictions outside** are **penalized** (red).

A simple idea is:

1. Dilate the ground truth image by a fixed kernel to form a "reward mask" (Figure 13(c))
2. For each pixel, count a pointwise loss:
    - White prediction on green mask: negative loss (reward)
    - White prediction on red mask: positive loss (penalty)
    - Black prediction: not rewarded or penalized
3. The total loss is the average loss across all pixels

The overlay Figure 13(d) is computed by the **pointwise multiplication** of Figure 13(b) and (c) ("matrix dot product")

Most pixels are rewarded in green. Only those far away from any true trace fall in red (penalty). Note that missed pixels (false negatives) are simply unrewarded—they incur no penalty beyond not being in the green zone.

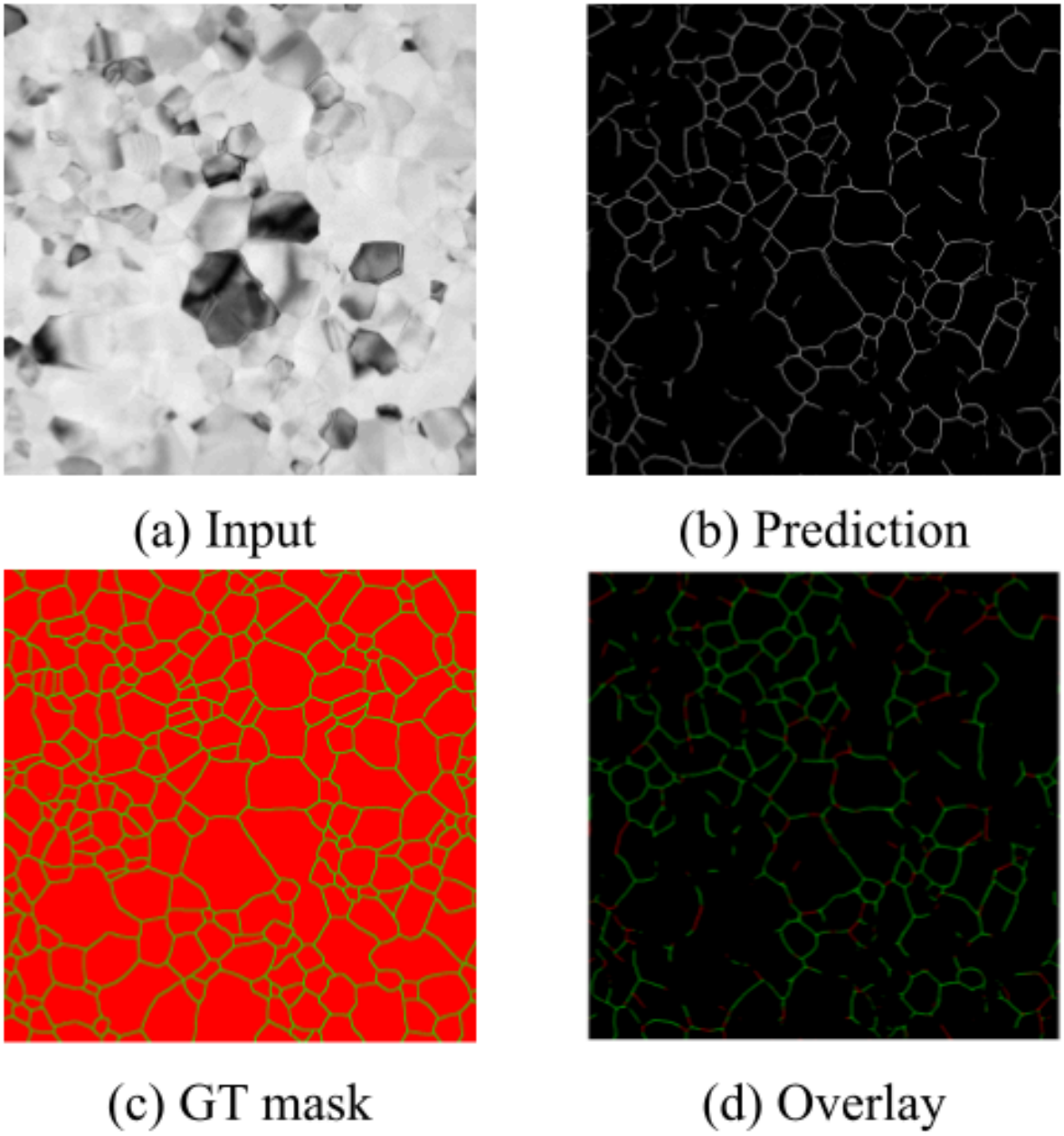


Figure 13: (a) input, (b) prediction, (c) ground-truth mask, and (d) overlay of prediction on ground truth. Results are from inference using a model trained for 300 epochs with an output resolution of 1024.

Result

The Top-hat loss function was tested on 3 example images at different FOVs, using a 3x3 dilation kernel on the ground-truth masks.

From Figure 14, across increasing epochs, the measured top-hat loss consistently decreases, indicating a steadily improvement alignment with the reference tracings. Furthermore, the loss level at about 60 epochs, consistent with human observation.

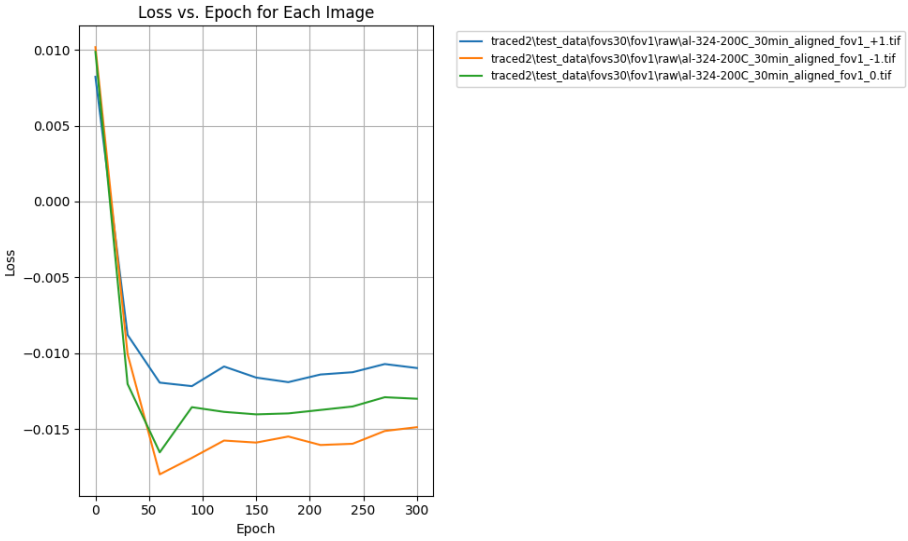


Figure 14. Loss vs. Epoch for the Top-hat loss function (kernel size = 3).

In Figure 15, the truth kernel size was changed to (5, 5). The resulting curve had a similar converging trend.

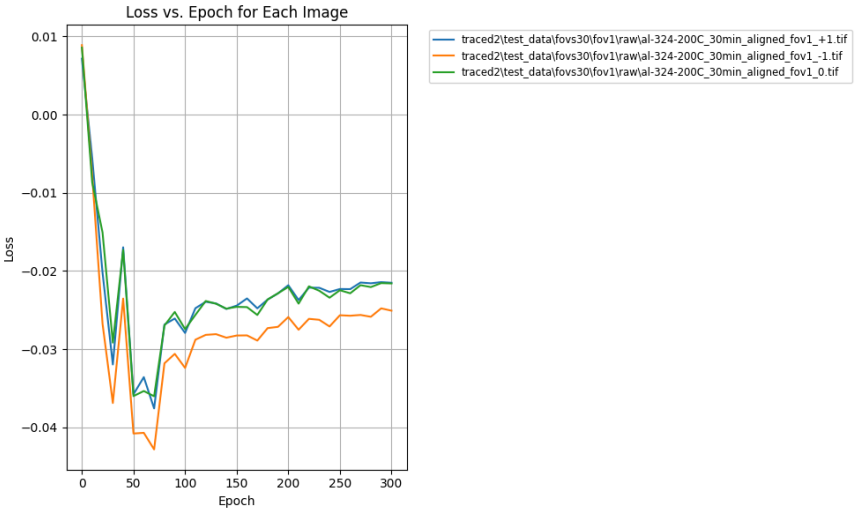


Figure 15. Loss vs. Epoch for the Top-hat loss function (kernel size = 5).

4. Normalized Gaussian Classification

In the previous section, we applied a fixed-pixel dilation to the ground-truth tracing and used a simple binary scheme—predictions inside the dilated mask earn a reward, and those outside incur a penalty.

In this section, we add another layer of complexity by introducing a continuous loss function that smoothly interpolates between reward and penalty: outputs closer to the true tracing receive higher rewards, while those farther away are penalized more heavily.

4.1 Procedure

1. The ground truth image is first inverted to white pixels on a black background
2. For each pixel, a Gaussian kernel (Figure 16 (a)) is applied. Essentially, each pixel is "smeared" to its neighborhood. There are two parameters for the Gaussian kernel:
    - Standard deviation σ: controls how far each pixel's influence spreads
    - Kernel size k: Truncates infinite Gaussian to a finite window since values beyond k*k become negligible
3. For each pixel, compute the Gasussian-weighed contributions of itself and its neighbors.

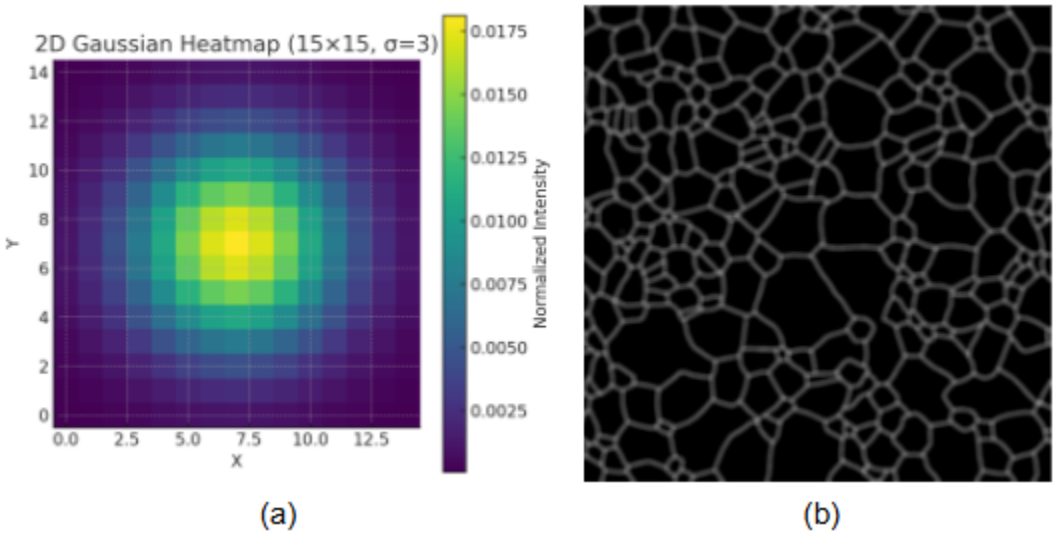


Figure 16. (a) Gaussian kernel of $k = 15$ and $\sigma = 3$; (b) The ground truth tracing dilated with the kernel (a).

Figure 16(b) illustrates that summing the Gaussian-weighted neighborhood makes intersections unnaturally bright—those pixels simply have more overlapping contributions. Figure 17(a) shows a local detail.

To correct this, we replace the sum with a **maximum** over the k*k window, so every pixel's value is the highest neighbor response rather than the sum. This is called **max-aggregation**, yielding uniform line intensity (Figure 17(b))

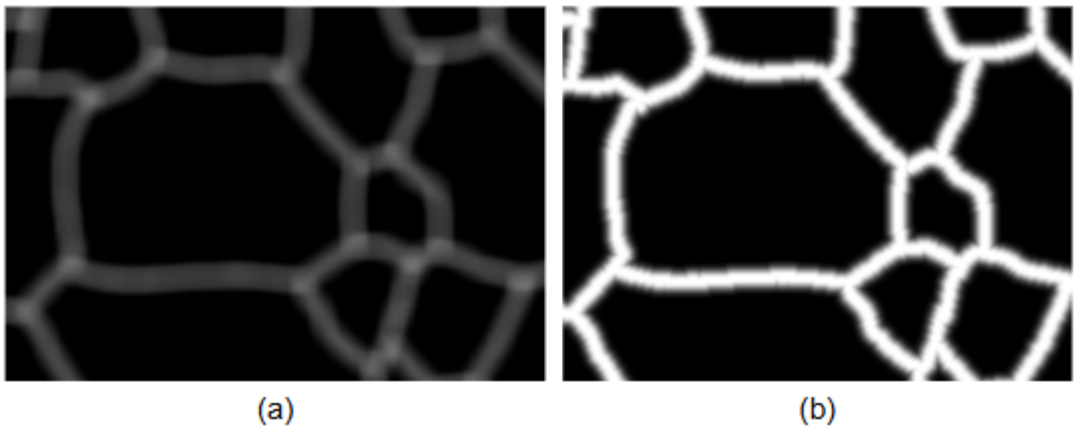


Figure 17. (a) Ground truth tracing dilated with a regular Gaussian kernel; (b) Gaussian dilation with max-aggregation and normalization.

4. Finally, all values are normalized to [0, 1], the same as the original image.

4.2 Reward Mask

The tracing dilated under the normalized Gaussian is then remapped from [0, 1] to [-1, 1].

In Figure 18, values near +1 appear green, values near -1 appear red, and zero appears black. This mirrors the top-hat scheme but now offers a smooth, turnable transition across categories, producing a green-black-red gradient with adjustable size and sharpness.

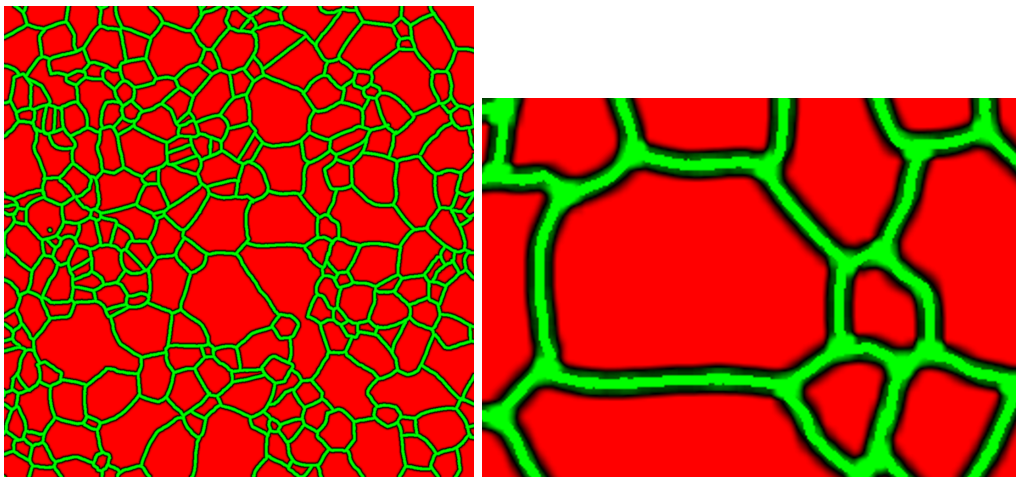


Figure 18: Gaussian dilated tracing mapped to a reward mask.

4.3 Computing Loss

Now, the loss function can be computed from a pointwise multiplication between the model output (Figure 19(b)) and the reward map (Figure 19(c)). Each foreground pixel (white, value = 1) is multiplied by its corresponding reward value, while background pixels (black, value = 0) contribute zero. Averaging over this product then gives the final loss.

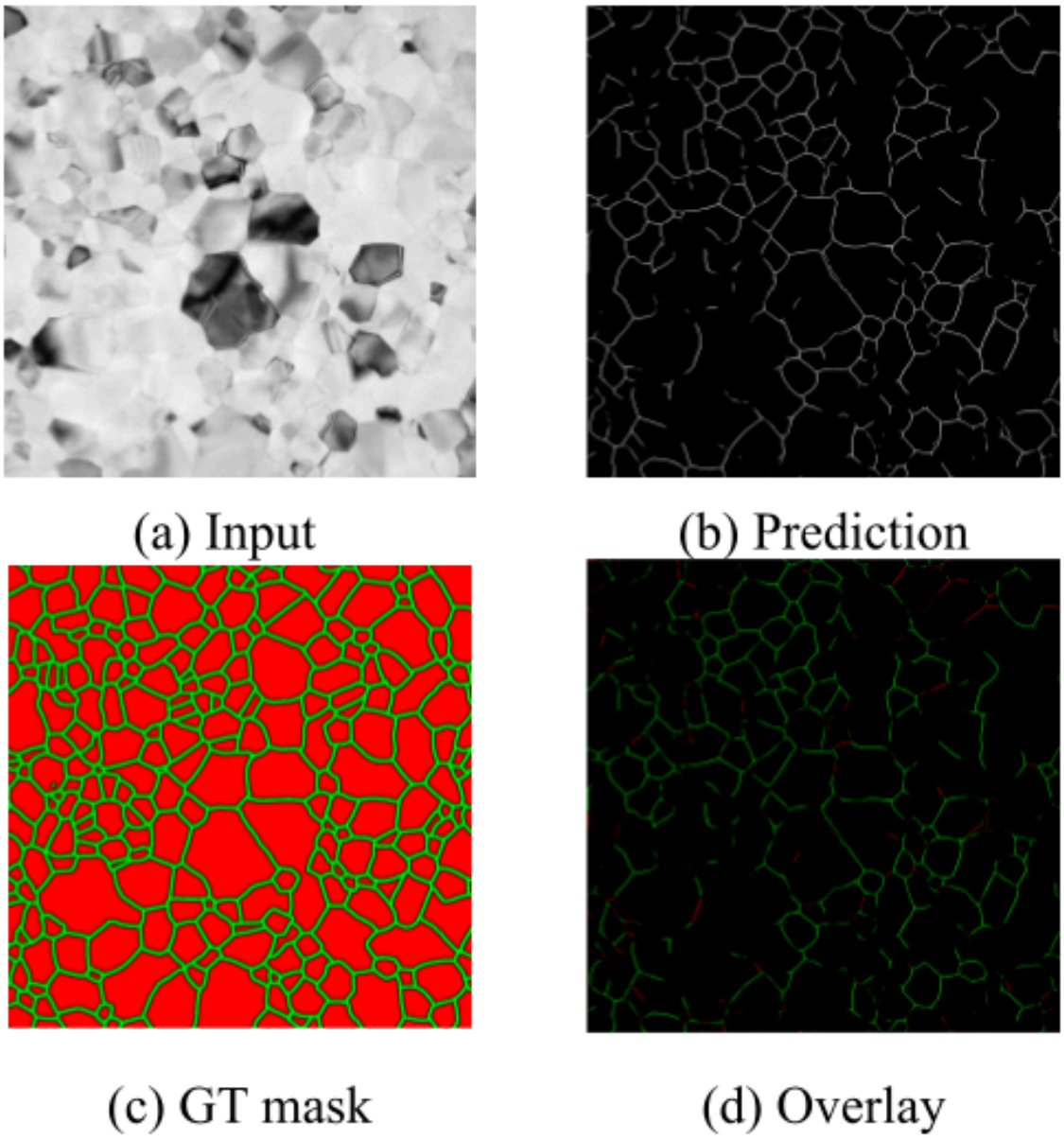
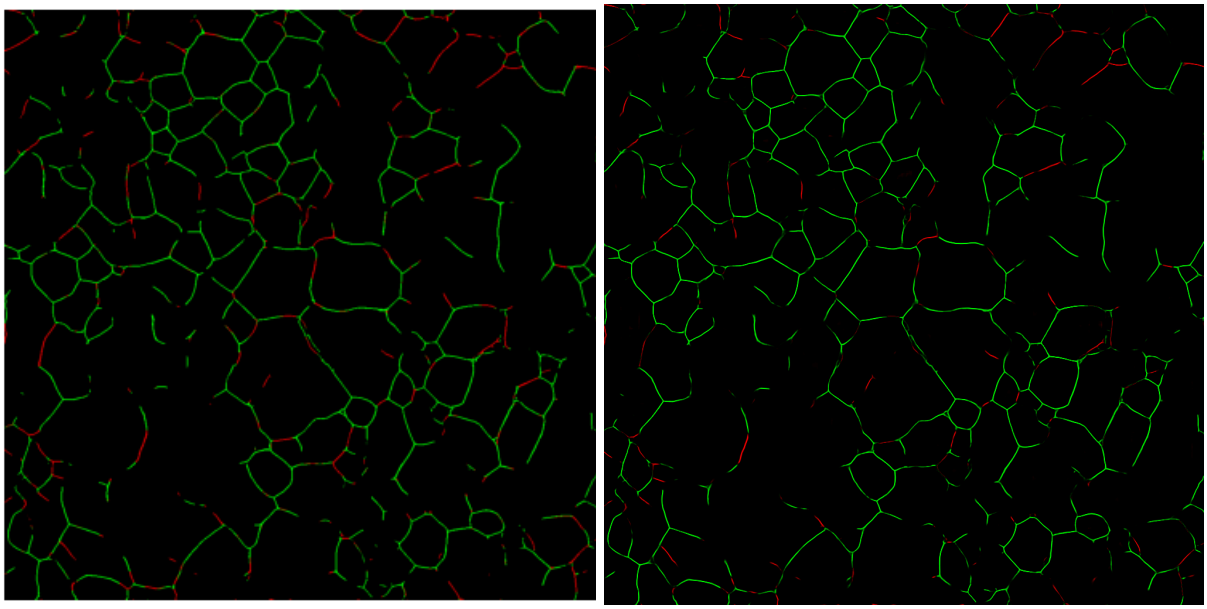


Figure 19. (a) input; (b) 300-epoch model prediction at resolution 1024; (c) Gaussian mask with $\sigma = 3$ and $k = 15$; (d) the resulting overlay.

As in Figure 19(d), correctly predicted pixels appear in green, slight misalignments fade through black, and complete errors appear in red.

4.4 Comparison with Top-Hat Filter



Above are the composite images for the Top-hat (Figure 13(d)) and normalized Gaussian (Figure 19(d)) filters.

- **Left**: the binary top-hat result sharply divides correct (green) from incorrect (red) predictions with no middle ground.
- **Right**: the smooth Gaussian filter

Both maps identify good and bad predictions similarly; the Gaussian approach is more forgiving of small offsets by offering a smooth transition between true and false positives.

4.5 Hyperparameters

Similar to the kernel size of the top-hat dilation, The Gaussian filter has a turnable hyperparameter: σ. A larger σ produces a wider spread, dilating the reward region more and yielding a more forgiving loss.

- Figure 19 uses the results for $\sigma = 3$, which we have been using before:
- Figure 20 reevaluates the same ground truth reward mask but on the output for a 50-epoch model.
- Figures 21 and 22 performed the same procedure but with $\sigma = 1$ for the ground truth reward mask.

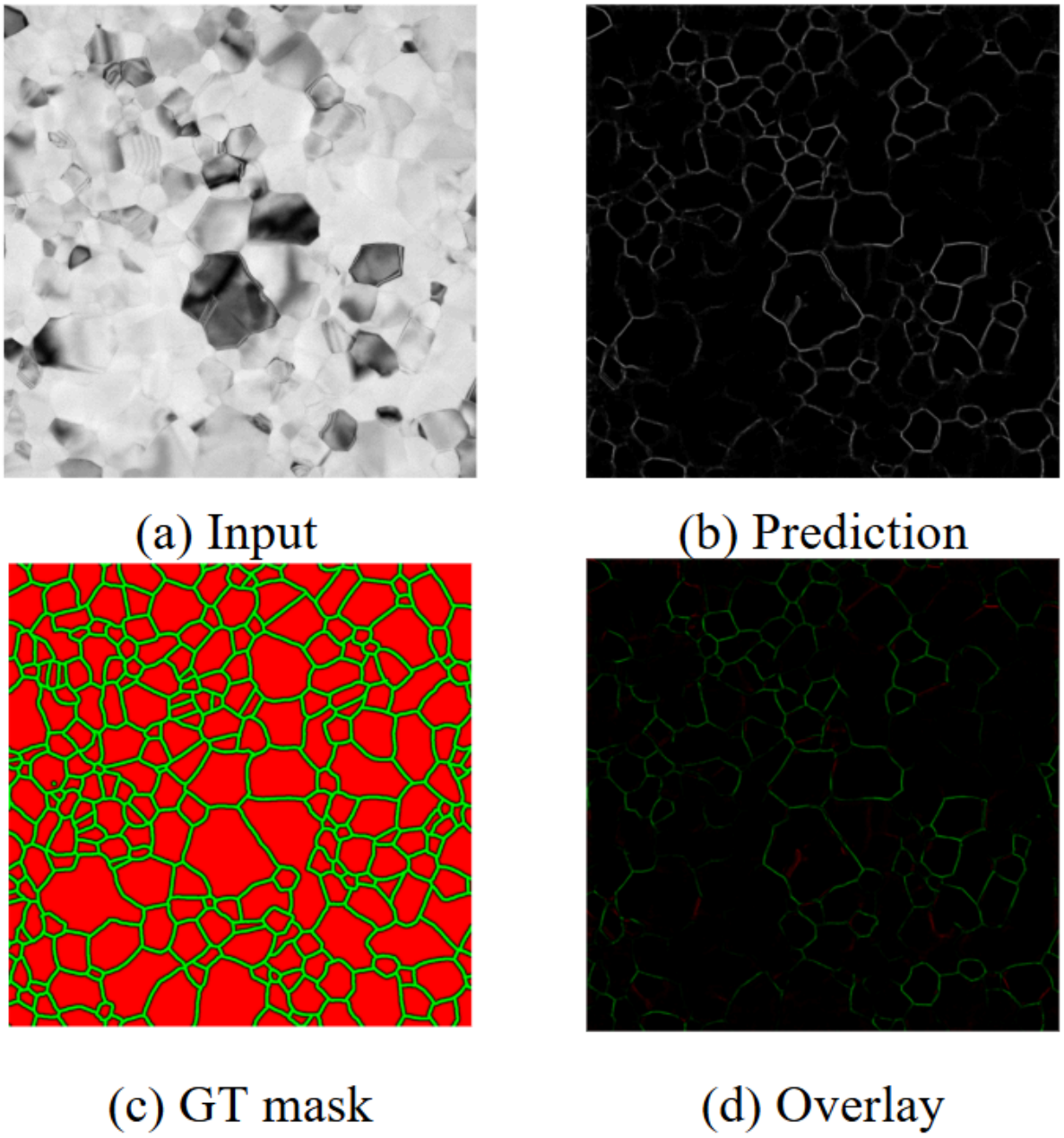


Figure 20. (a) input; (b) **50**-epoch model prediction at resolution 1024; (c) Gaussian mask with $\sigma = 3$ and $k = 15$; (d) the resulting overlay.

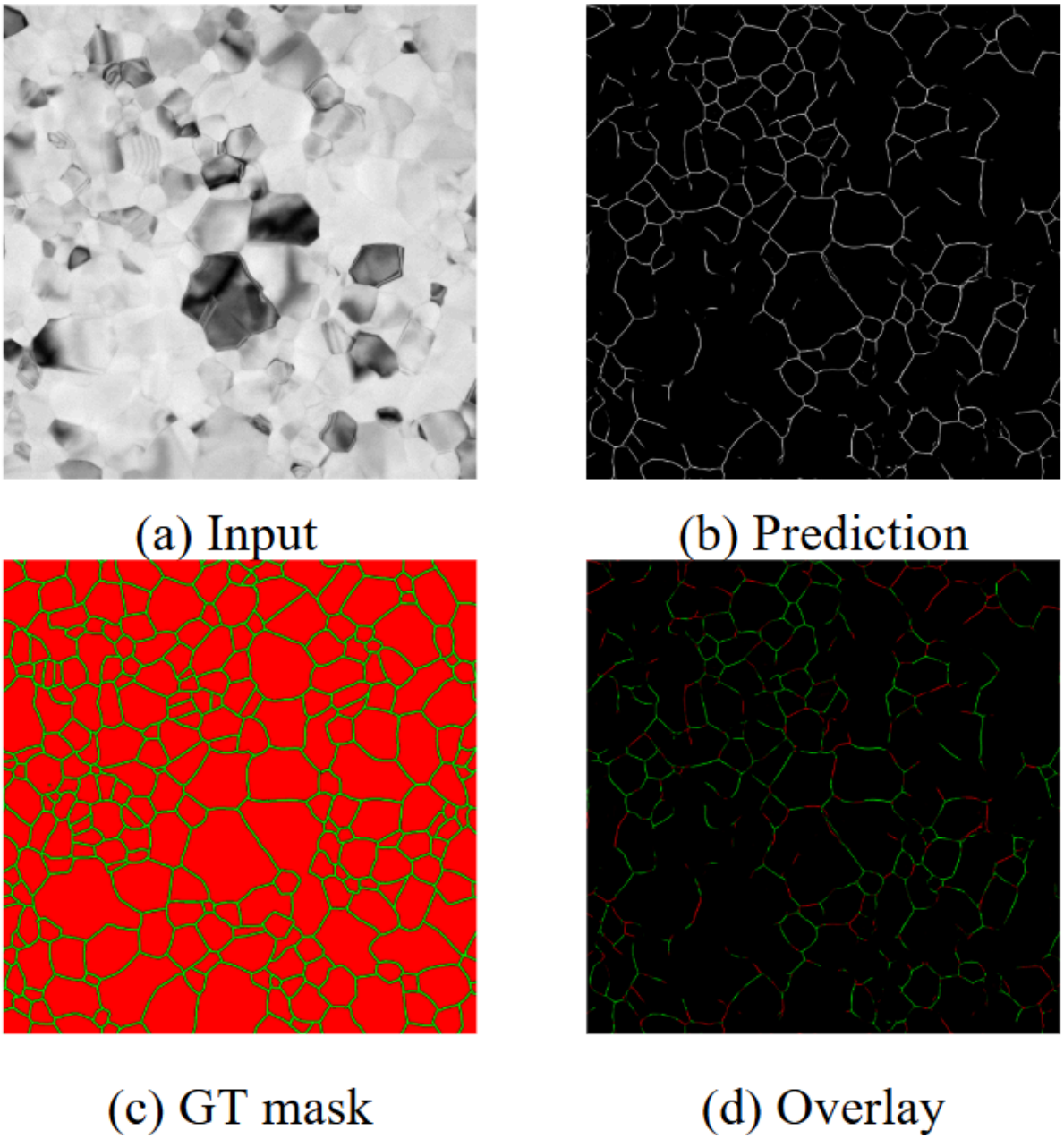


Figure 21. (a) input; (b) 300-epoch model prediction at resolution 1024; (c) Gaussian mask with $\boldsymbol{\sigma} = \mathbf{1}$ and $k = 15$; (d) the resulting overlay.

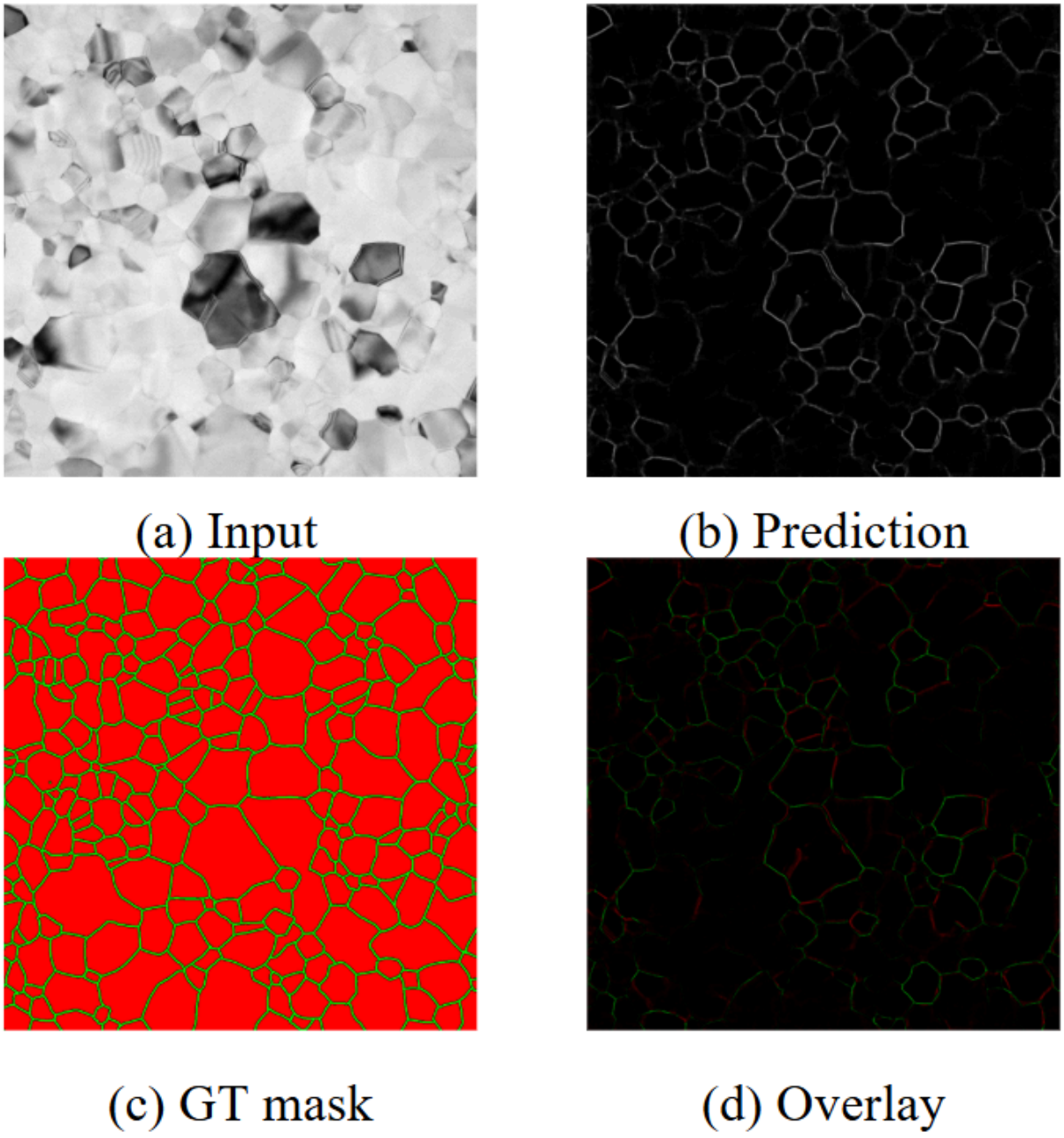


Figure 22. (a) input; (b) **50**-epoch model prediction at resolution 1024; (c) Gaussian mask with $\boldsymbol{\sigma = 1}$ and $k = 15$; (d) the resulting overlay.

The result is summarized in Table 1. With a smaller value of σ, the loss increases for both 50 epochs and 300 epochs due to a stricter metric.

Normalized Gaussian Loss	σ = 3	σ = 1
50 Epochs	-5.8e-3	+6.9e-3
300 Epochs	-17.9e-3	-4.7e-3

Table 1. Normalized Gaussian loss function under different σ hyperparameters and model epochs.

4.6 Loss vs Epoch

We evaluated the Gaussian‑filtered loss (σ = 3) on UNet models saved at epochs 0 through 300. The resulting loss curve decreases monotonically, exactly matching the qualitative observations of steadily improving predictions as training progresses.

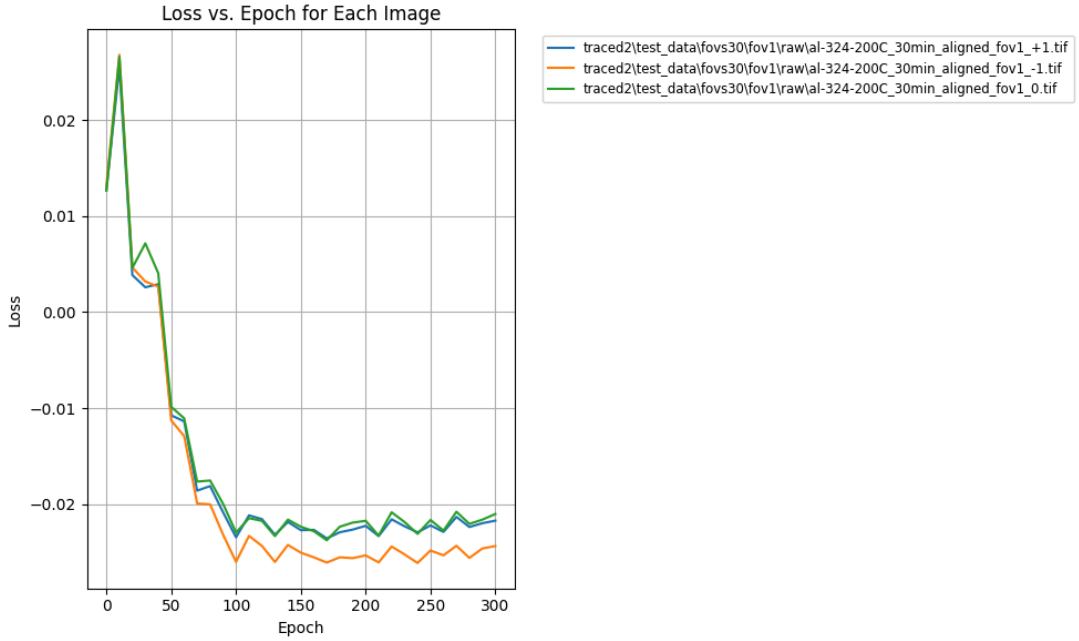


Figure 23. Loss vs epoch for the normalized Gaussian loss function, with σ = 3

The normalized Gaussian dilation to the tracing image

5. Future Work

This loss function is far from complete. Future works include:

- Make a balanced penalty for false negative predictions. Since the positive/negative classes are imbalanced, a proper weight must be applied.
  This will yield a function similar to BCE but with spatial tolerance baked in.
- Explore other non-pointwise loss functions.
- Adapt this loss function to Pytorch. This can be used in conjunction with traditional loss functions, such as BCE.

Conclusion

In this report, we tackled two challenges in grain-boundary detection. First, we developed a structured data-augmentation pipeline—leveraging systematic GIMP-based cropping and a clear naming convention—to expand our training set while preserving precise input–mask alignment. Second, we identified the limitations of standard BCE loss for misalignment-sensitive tasks and introduced a novel family of spatially-aware metrics, from binary top-hat dilation to a continuous Gaussian-dilated reward map, that better correlates with visual tracing quality. Our experiments on UNet models (0–300 epochs) show that the Gaussian-filtered loss decreases monotonically and plateaus in concert with qualitative convergence, demonstrating its value as both an evaluation tool and a potential training objective. Going forward, integrating balanced penalties for false negatives, exploring holistic shape metrics, and embedding this loss directly into PyTorch will further strengthen model robustness and alignment with human judgment.

Code and Figures

The code for this project can be found on the `ming-loss-grain` branch of the UNet GitHub repository: https://github.com/MatthewJPatrick/grain_unet_working/tree/ming-loss-grain

The figures in this report are in this slideshow

Acknowledgment

I would like to express my sincere gratitude to Matthew Partick, Professor Katayun Barmak, and Lauren Grae for their invaluable guidance on the project's framework, tools, and details.