

Thomas Kidd & Zach Wilson

Collaborative Image Denoising with Deep Learning



Problem & Motivation

01 Why denoising matters

01 Photography, medical, drone footage

02 The Real World challenge

02 Blur, motion, and noise in real-world datasets

03 Goal: Develop and compare multiple deep learning approaches for denoising

03 Plan: Compare Architectures using two datasets and evaluating across both

DATASET OVERVIEW



GOPRO

10,000 real-world motion blur images with sharp-ground truth pairs, captured using high-speed handheld cameras for realistic deblurring benchmarks.

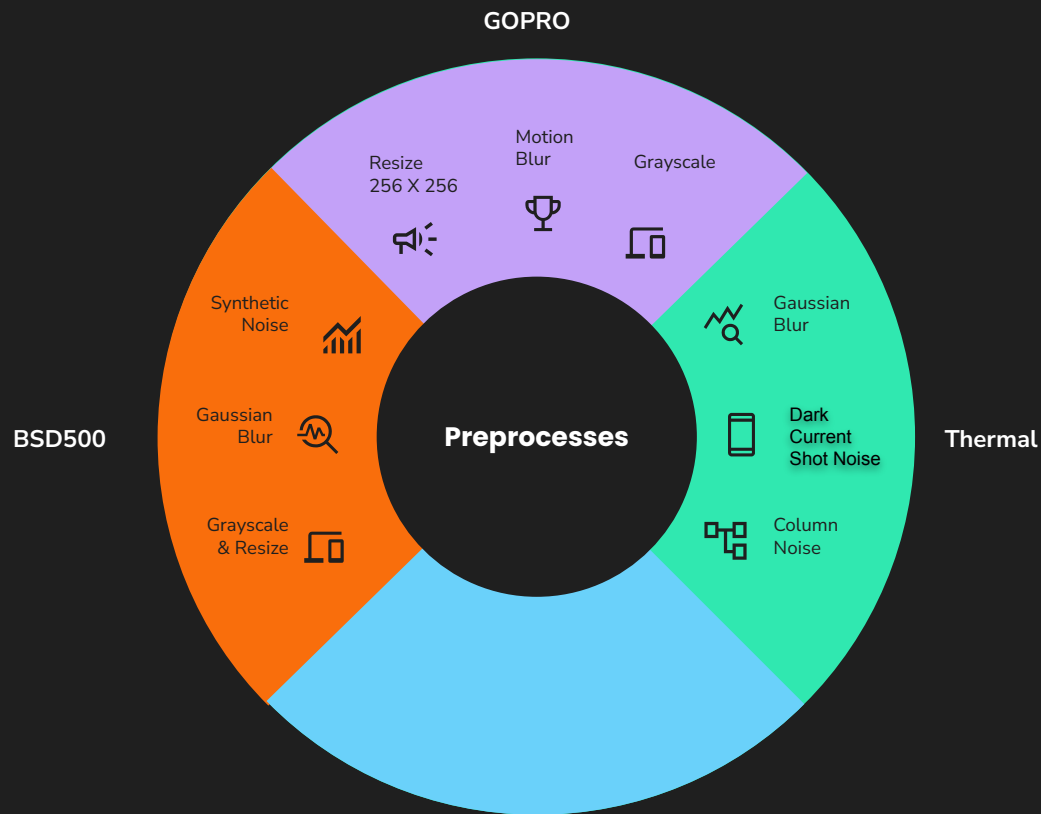
BSD500

Berkeley segmentation dataset of 500 natural images from commonly used for benchmarking image segmentation and denoising algorithms due to its detailed textures and ground truth annotations.

Thermal

PBVS Thermal Dataset for super resolution was used to create a noisy thermal image dataset. High resolution images were disturbed with noise to create realistic noisy data.

Dataset Preprocessing

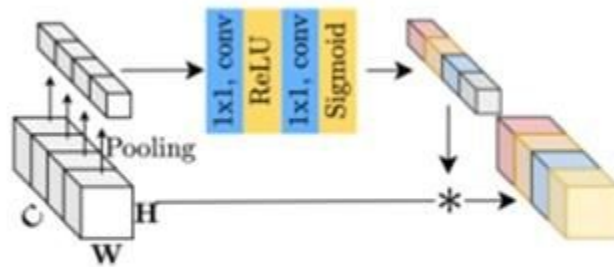


Architecture Comparison

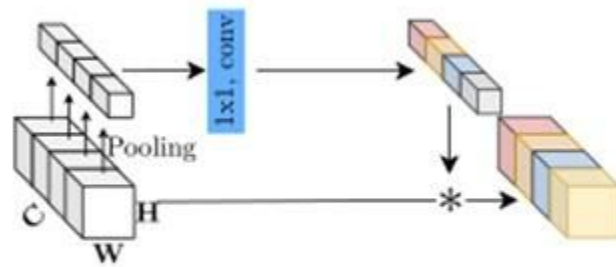


Feature	U-Net	NAFNET
Framework	Tensorflow	Pytorch
Architecture	Deep U-Net Autoencoder	U-Net Style
Input Size	256 X 256 grayscale	256 X 256 grayscale
Special Features	Cosine LR, AdamW, EarlyStopping	Nonlinear activation free blocks

Architecture Comparison



(a) Channel Attention



(b) Simplified Channel Attention

Deep U-Net Autoencoder

```
# Encoder
c1 = layers.Conv2D(64, 3, activation='relu', padding='same')(inputs)
c1 = layers.Conv2D(64, 3, activation='relu', padding='same')(c1)
p1 = layers.MaxPooling2D((2, 2))(c1) # 128x128

c2 = layers.Conv2D(128, 3, activation='relu', padding='same')(p1)
c2 = layers.Conv2D(128, 3, activation='relu', padding='same')(c2)
p2 = layers.MaxPooling2D((2, 2))(c2) # 64x64

c3 = layers.Conv2D(256, 3, activation='relu', padding='same')(p2)
c3 = layers.Conv2D(256, 3, activation='relu', padding='same')(c3)
p3 = layers.MaxPooling2D((2, 2))(c3) # 32x32

c4 = layers.Conv2D(512, 3, activation='relu', padding='same')(p3)
c4 = layers.Conv2D(512, 3, activation='relu', padding='same')(c4)
p4 = layers.MaxPooling2D((2, 2))(c4) # 16x16

# Bottleneck
bn = layers.Conv2D(1024, 3, activation='relu', padding='same')(p4)
bn = layers.Conv2D(1024, 3, activation='relu', padding='same')(bn)

# Decoder
u4 = layers.UpSampling2D((2, 2))(bn)
u4 = layers.Concatenate()([u4, c4])
c5 = layers.Conv2D(512, 3, activation='relu', padding='same')(u4)
c5 = layers.Conv2D(512, 3, activation='relu', padding='same')(c5)

u3 = layers.UpSampling2D((2, 2))(c5)
u3 = layers.Concatenate()([u3, c3])
c6 = layers.Conv2D(256, 3, activation='relu', padding='same')(u3)
c6 = layers.Conv2D(256, 3, activation='relu', padding='same')(c6)

u2 = layers.UpSampling2D((2, 2))(c6)
u2 = layers.Concatenate()([u2, c2])
c7 = layers.Conv2D(128, 3, activation='relu', padding='same')(u2)
c7 = layers.Conv2D(128, 3, activation='relu', padding='same')(c7)

u1 = layers.UpSampling2D((2, 2))(c7)
u1 = layers.Concatenate()([u1, c1])
c8 = layers.Conv2D(64, 3, activation='relu', padding='same')(u1)
c8 = layers.Conv2D(64, 3, activation='relu', padding='same')(c8)

outputs = layers.Conv2D(1, 1, activation='sigmoid', padding='same')(c8)
```

This architecture extends the classic U-Net with a deeper encoder-decoder structure to capture complex spatial features.

Encoder: Four levels of convolutional blocks, each with two Conv2D layers followed by MaxPooling2D. These progressively reduce spatial dimensions while increasing feature depth, enabling hierarchical feature extraction.

Bottleneck: Two Conv2D layers with 1024 filters to capture the most abstract features before upsampling begins.

Decoder: Each upsampling step is followed by skip connections from the encoder and two Conv2D layers. These skip connections preserve spatial context and details lost during downsampling.

Output Layer: A final Conv2D layer with sigmoid activation to produce the denoised grayscale image.

Deep-U Loss and Metrics

& NAFNET

Using a hybrid loss function we average the Mean Absolute error and Mean Squared Error to give a balanced and stable optimization target

```
def combined_loss(y_true, y_pred):  
    mse = tf.reduce_mean(tf.square(y_true - y_pred))  
    mae = tf.reduce_mean(tf.abs(y_true - y_pred))  
    return 0.5 * mse + 0.5 * mae
```

MSE+ MSA = Noise Robust +
Robust to Outliers

L1-based composite loss function that combines a diffusion model component and a pixel-wise component

```
if config.LOSS == 'L1':  
    self.loss = nn.L1Loss()  
elif config.LOSS == 'L2':  
    self.loss = nn.MSELoss()
```

L1 = pixel-wise loss

Peak Signal to Noise Ratio (PSNR) measures pixel reconstruction accuracy with higher values indicating better denoising

```
def psnr_metric(y_true, y_pred):  
    return tfi.psnr(y_true, y_pred, max_val=1.0)
```

PSNR = pixel-wise similarity

Structural Similarity index (SSIM) measures perceptual similarity, with “1” being optimal. Which is a better indicator of denoising than loss

```
def ssim_metric(y_true, y_pred):  
    return tfi.ssim(y_true, y_pred, max_val=1.0)
```

SSIM = structural and perceptual
fidelity

Training Setup

Deep-U vs NAFNET

Optimizer – AdamW:

AdamW is an improved variant of Adam that decouples weight decay from the gradient update, leading to better generalization, especially in deep networks like UNet.

Learning Rate – Cosine Decay:

The learning rate gradually decreases following a cosine curve from an initial high value to a small value. This helps the model converge smoothly and avoids overshooting, especially in later training stages.

EarlyStopping + ModelCheckpoint:

We monitor validation loss:

- **EarlyStopping** halts training if no improvement is seen for 5 epochs, saving time and preventing overfitting.
- **ModelCheckpoint** ensures we retain the weights from the best epoch (lowest validation loss), giving us the best-performing model.

Optimizer – AdamW:

- Weight decay = $1e-4$
- Implements weight decay regularization to prevent overfitting
- Adaptive learning rate adjustment based on parameter history

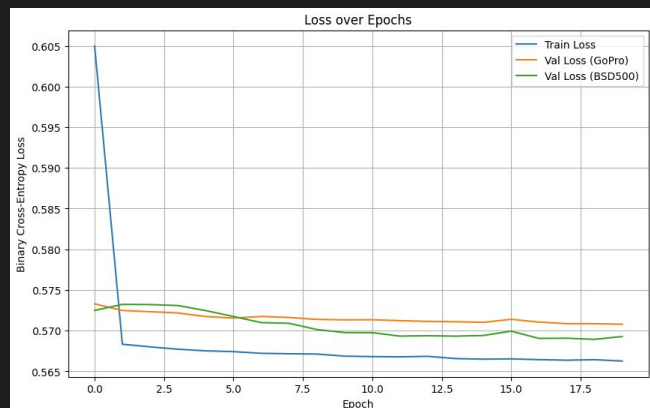
Learning Rate – 0.0001 ($1e-4$)

- Static learning rate throughout training
- Balanced for stable convergence without oscillation

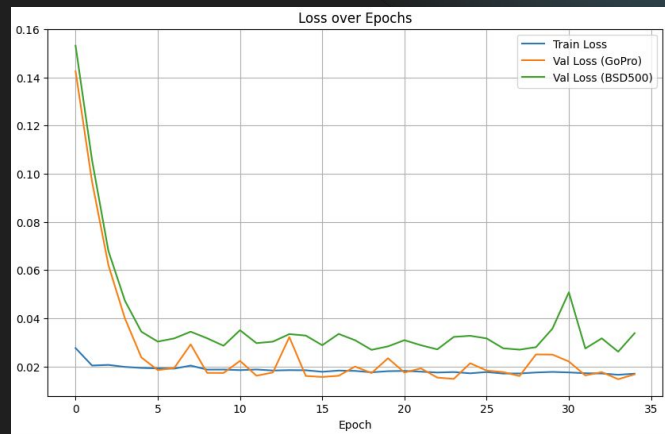
ModelCheckpoint:

- Saves models at regular intervals (every 10,000 iterations)
- Maintains "best model" based on PSNR validation metric
- Validation performed every 1,000 iterations
- Tracks multiple image quality metrics (PSNR, SSIM, LPIPS, DISTs)
- Maximum training duration: 100,000 iterations

Training Results



Binary Cross-Entropy Loss without learning rate scheduler

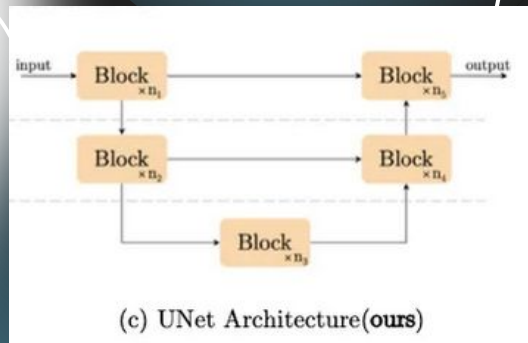
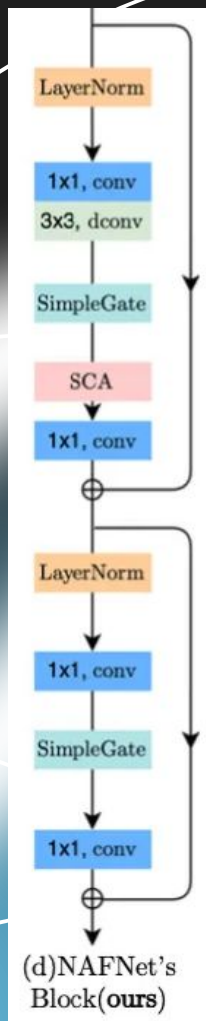


Hybrid MAE+MSE average loss with cosine learning rate scheduler

PSNR (GoPro): 25.83, SSIM (GoPro): 0.8084
PSNR (BSD500): 23.01, SSIM (BSD500): 0.5117
Test on GoPro:
70/70 1s 10ms/step - loss: 0.5811
Test on BSD500:
13/13 0s 27ms/step - loss: 0.5858

PSNR (GoPro): 29.62, SSIM (GoPro): 0.8900
PSNR (BSD500): 24.33, SSIM (BSD500): 0.6319
Test on GoPro:
70/70 6s 79ms/step - loss: 0.0096 - psnr_metric: 31.3383 - ssim_metric: 0.9226
Test on BSD500:
13/13 2s 149ms/step - loss: 0.0226 - psnr_metric: 24.4270 - ssim_metric: 0.6290

NAFNET Network



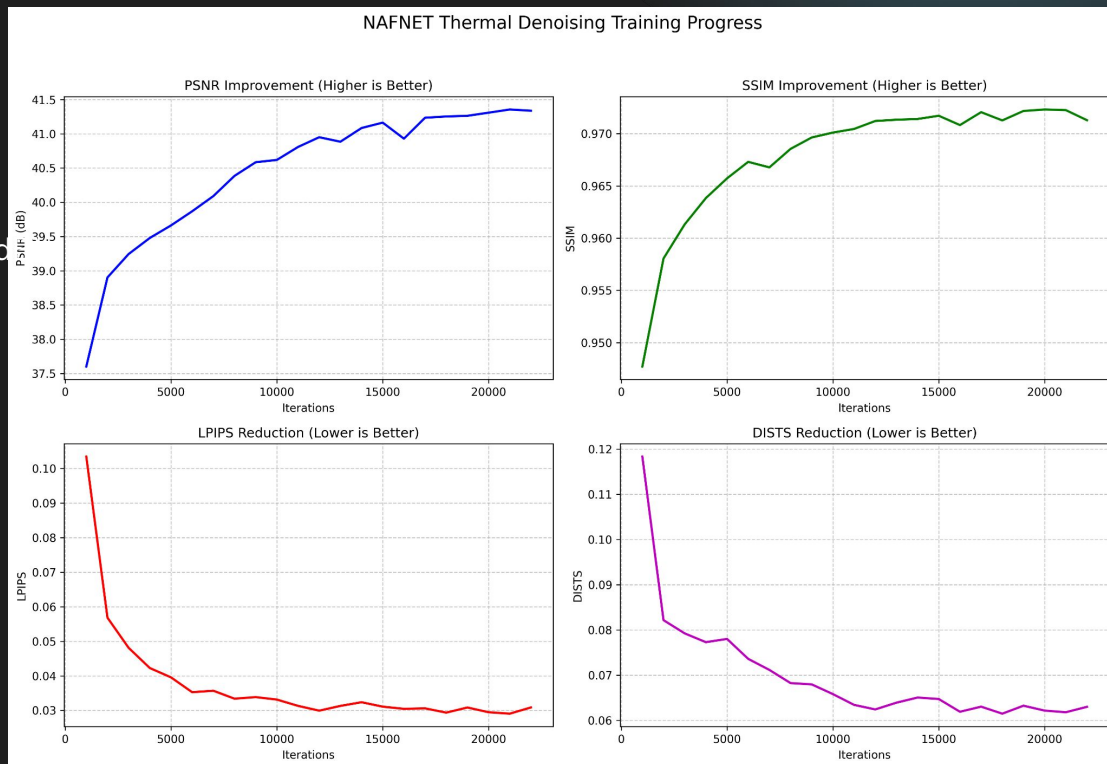
NAFNet: Efficient Image Restoration (Deblurring/Denoising)

- **Core Block:** Simplified unit with Layer Norm, two Conv layers, a "Simple Gate" (split-channel modulation for activation-free refinement), & skip connection.
- **Architecture (U-Net Style):**
 - **Encoder:** NAFNet blocks for multi-scale feature extraction via downsampling.
 - **Bottleneck:** Processes lowest-resolution abstract features.
 - **Decoder:** NAFNet blocks for upsampling and feature reconstruction.
 - **Skip Connections:** Link encoder-decoder for detail preservation.
- **Key: Nonlinear Activation Free:** No traditional activations (ReLU, etc.); Simple Gate provides nonlinearity, simplifying & boosting performance.
- **Output:** Final Conv layer reconstructs the restored image. ¹¹




NAFNET Training Results

Training Progress Overview:

PSNR: 37.60 dB \rightarrow 41.34 dB (Improvement: 3.73 dB)
SSIM: 0.9477 \rightarrow 0.9713 (Improvement: 0.0236)
LPIPS: 0.1035 \rightarrow 0.0309 (Reduction: 0.0726)
DISTS: 0.1183 \rightarrow 0.0630 (Reduction: 0.0553)



Training and Evaluation Results

Network		GoPro Loss	BSD500 Loss	PSNR GoPro/BSD500	SSIM GoPro/BSD500	PSNR/SSIMThermal
	Deep-UNet Hybrid loss	0.0096	0.0226	31.33/24.42	0.9226/0.6290	41.35 / 0.9723
	Deep-UNet Binary Cross-Entropy	0.0139	0.0342	27.25/21.99	0.8531/0.5820	N/A
	NAFNET	NA	NA	37.22	0.965	41.34 / 0.9713

BEST MODEL SUMMARY



NAFNET

Noisy



Clean



Deblurred



Deblurred



DEEP-U

Noisy



Clean



Deblurred



Deblurred



NAFNET



DEEP-U



Image: 028_01_D4_th.bmp (ROI at x=491, y=80)

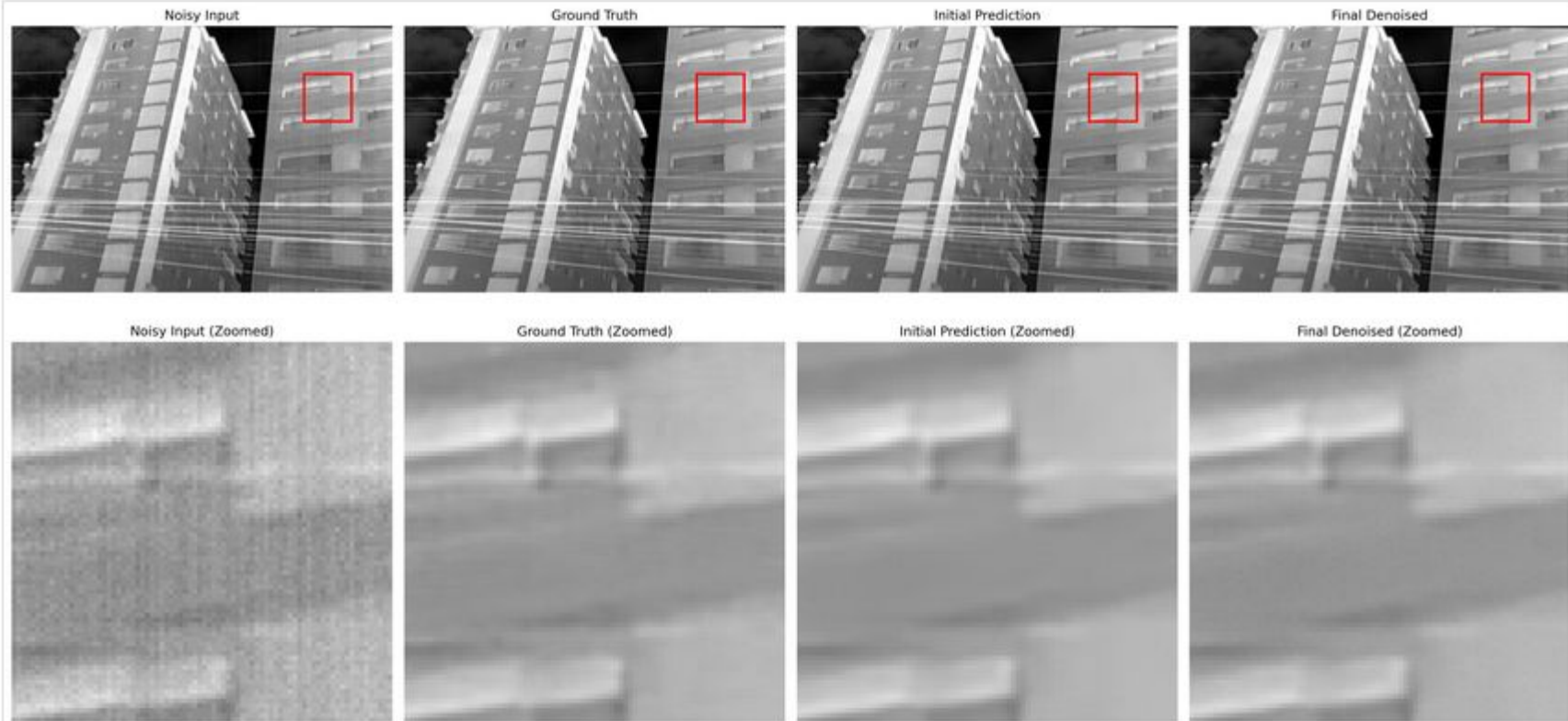
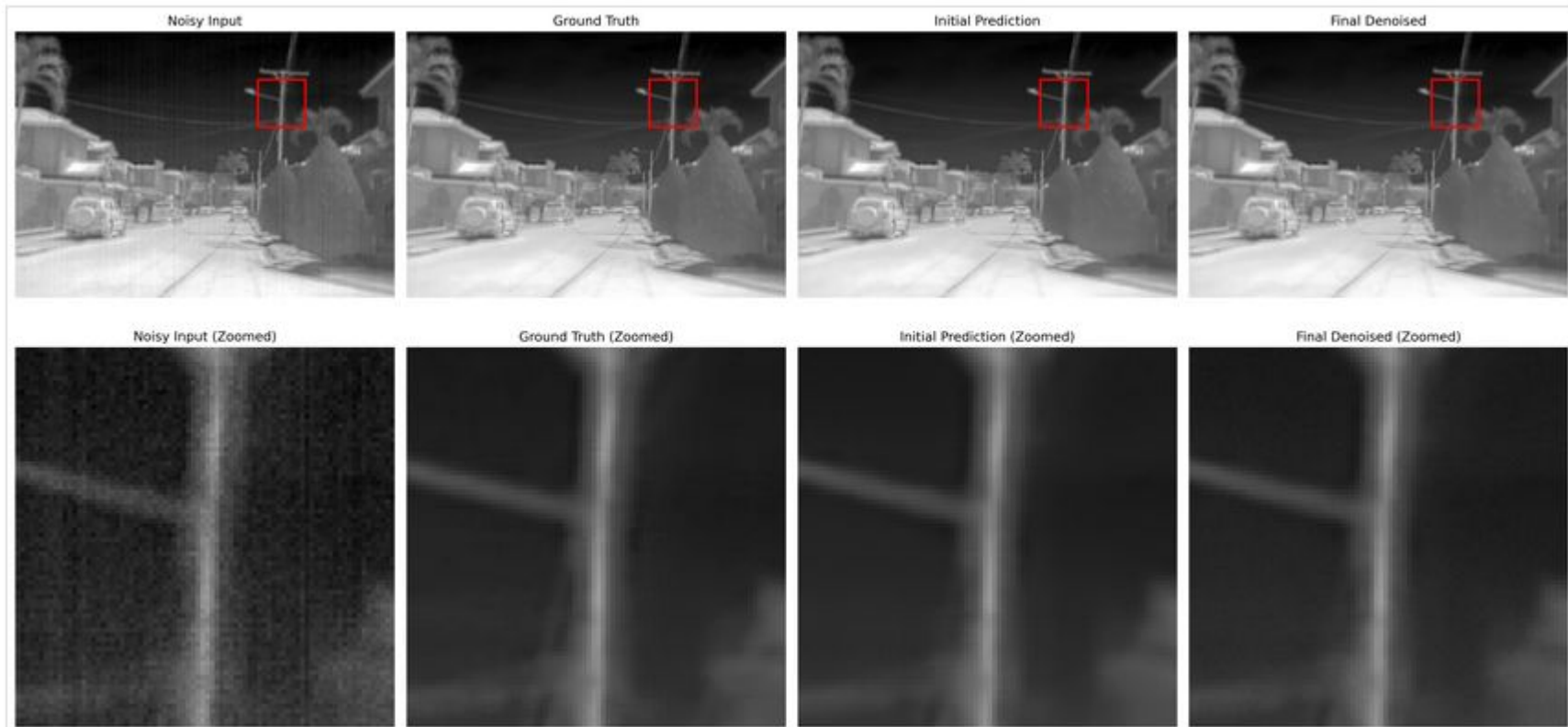


Image: 044_01_D1_th.bmp (ROI at x=409, y=80)



Next Steps

01 Focus on deblurring methods

02 DeblurGAN-v2

03 Restormer

Questions?

Two decorative teal lines, one slightly above the other, curving under the word "Questions?".Two large, thin white arcs in the top right corner of the slide, creating a modern, abstract design.