Reducing Noise Artifacts in Boundary Detection for Low SNR Images

Thomas Snow, Yuchong Zhang

December 7, 2024

Abstract

We introduce a method for improving the boundary detection results of existing models at low SNR. Our method applies cross-patch attention over a collection of noisy boundary results to distinguish the true boundary from the noise artifacts by learning from the structural features of the noisy boundary detection results. Furthermore, our method can be adapted to perform image denoising and it produces good results at multiple noise levels.

1 Introduction

Boundary detection/extraction is a fundamental problem within computational imaging and computer vision, with applications spanning autonomous driving cars, robotics, and medical imaging. The growing capabilities of machine learning have produced boundary detection methods with significantly improved performance and efficiency. Since the early work of Cranny [Can86], we have seen a plethora of boundary detection algorithms and models that perform near-perfectly on inputs with high signal-to-noise-ratio (SNR). A natural extension is to perform boundary detection on noisy images (low SNR), where the boundary structure is obscured and difficult to distinguish. Many works focus on boundary extraction for low SNR inputs [PHH+24], [XLG24], [Ofi24]; however, these methods often introduce noise artifacts in the results or lose information about the true boundary structure. In this paper, we address this problem. We propose a method to reduce the noise artifacts while minimizing the loss of boundary information for images with low SNR.

As seen in figure 1, at increasing levels of noise the boundary attention model [PHH+24] produces *phantom edges* which are noise artifacts that the model mistakenly outputs as boundaries within the image. A key observation is that as noise is resampled or increases, these artifacts undergo drastic changes while the correct boundary remains. Furthermore,

[1]Implementation of our method can be found at `https://github.com/yc7z/ClearBoundary`

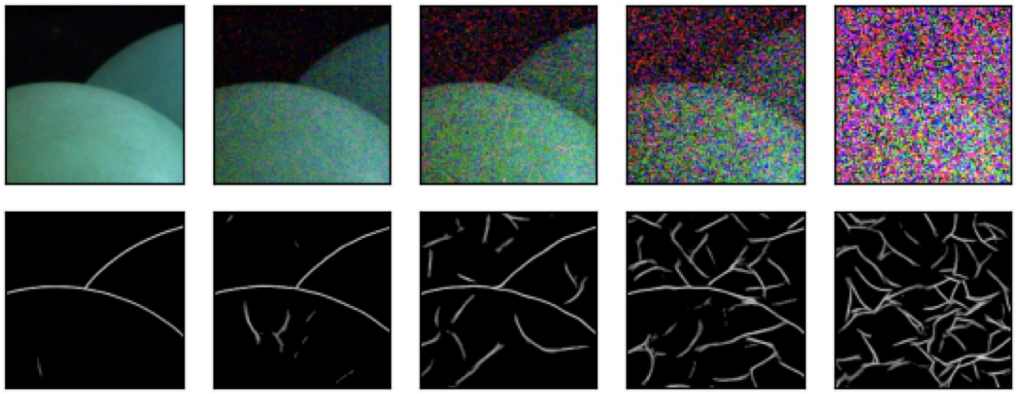

Figure 1: Boundary Attention result as noise level increases (SNR decreases) from [PHH+24].

we see that at higher levels of noise, the correct boundary becomes obscure. However, one can still make out its general shape. Intuitively, when one takes multiple noisy samples, the general shape of the correct boundary will remain while the noise artifacts will differ greatly across samples. With this in mind, we propose the following model to differentiate the true boundary from the noise artifacts while correcting the obscurities by taking into consideration the predicted boundaries of all the noisy inputs.

We further note that with the rise of batch-based photography such as High-Dynamic-Range (HDR) [RBS00] the assumption of having multiple, potentially noisy, images is valid in practice, illustrating the need to develop batch-based methods to enhance photography.

Although designed for improving boundary detection, our method can easily be generalized to other types of denoising problems as seen in section 6.

2 Related Work

Boundary Detection at Low SNR

Building on the work of Verbin and Zickler [VZ21], who introduced the Field of Junctions (FoJ) method, significant advancements have been made in boundary detection for low-SNR images [PHH+24], [XLG24]. Both of these methods rely heavily on FoJ. At a high level, Field of Junctions (FoJ) first breaks an image into many overlapping small patches. Each patch then goes through an initialization phase which learns the position of the junction (if such a position exists) and fits the contours to best describe changes in color intensities. The next phase, called the refinement phase, consists of combining patches in a way that best matches the image structure. This partition of the pipeline into two phases is common within boundary detection at low SNR, Xu, Luo, and Guo [XLG24] use a similar approach by splitting their model into initialization and refinement stages. Their Initialization stage utilizes the FoJ methodology which is achieved through the use of Convolutional Neural Networks (CNN). In contrast, their refinement phase employs a transformer with multi-head

attention layers employed on a per-patch basis using global information regarding the FoJ construction of each patch.

The Vision Transformer

The transformer, first introduced in [VSP+17], was originally designed for language translation tasks. Integral to the architecture is the self-attention module, which allows the model to learn the dependencies between elements within the input sequence. [DBK+21] applies the transformer to computer vision tasks by dividing an image into small patches and using the patches to form the input sequences, thereby allowing the model to learn the spatial relationship across patches.

3 Methods

3.1 Setup

We first describe the intuition behind our approach. Given a collection $\mathcal{S} = \{S_1, \ldots, S_N\} \subset \mathbb{M}_{n\times n}([0,1])$ of noisy signals - in our case we will assume that such signals are 2-dimensional matrices with entries lying in the interval $[0,1] \subset \mathbb{R}$ (normalized images) - we break each respective signal into $M = m^2$ square patches of constant size with stride $\tau \in \mathbb{N}$. In particular, if $\tau = 0$, then the patches do not overlap, but if $0 < \tau < m$, then the patches will overlap, hence the $\approx$ below (if $\tau = 0$ this is equality).

$$S_i \approx \begin{bmatrix} P^{(i)}_{(1,1)} & \cdots & P^{(i)}_{(1,m)} \\ \vdots & \ddots & \vdots \\ P^{(i)}_{(m,1)} & \cdots & P^{(i)}_{(m,m)} \end{bmatrix}, \ \forall i = 1, \ldots, N.$$

For each $(i,j) \in [m] \times [m]$, let $B_{(i,j)} = \left\{P^{(1)}_{(i,j)}, \ldots, P^{(N)}_{(i,j)}\right\}$ denote the *batch* of $(i,j)^{th}$ patches from each input signal. Furthermore, let $\mathcal{B} = \left\{B_{(i,j)} \mid (i,j) \in [m] \times [m]\right\}$ denote the collection of batches produced by the collection of noisy signals, which will serve as input to our model. For training purposes, we also partition the clean/true signal $C \in \mathbb{M}_{n\times n}([0,1])$ in the same manner. Let $C_{(i,j)}$ denote the $(i,j)^{th}$ patch of the clean signal for all $(i,j) \in [m] \times [m]$.

Let $\mathcal{BA}$ denote the boundary attention model [PHH+24] which takes as input an image and produces an image of the same dimensions with the predicted boundaries. In our context, we are given a set of noisy images $\mathcal{I} = \{I_1, \ldots, I_N\}$. We use $\mathcal{BA}$ to produce our collection of noisy signals $\mathcal{S} = \{S_1, \ldots, S_N\}$, where $S_i = \mathcal{BA}(I_i)$ for all $i = 1, \ldots, N$.

There are two natural problem setups for which our method can be employed. Note that in this paper we consider Gaussian noise to simulate read noise; however, our method can be extended to any noise distribution. Let $\mathbb{M}_{n\times n}(\mathcal{N}(0,\sigma))$ denotes the space of $n \times n$ matrices with entries sampled from $\mathcal{N}(0,\sigma)$.

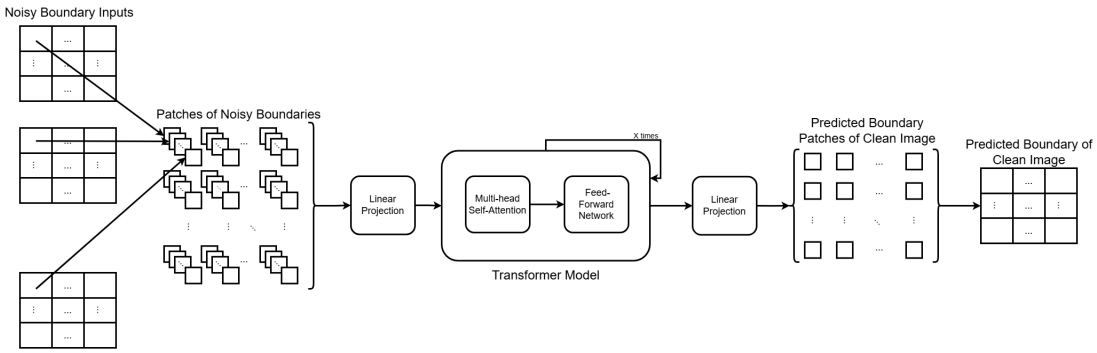

Figure 2: Architecture of our Model

Problem 1. *In this setup, we assume each $S_i \in \mathcal{S}$ is of the form $S_i = C + \mathbb{M}_{n\times n}(\mathcal{N}(0, \sigma_i))$, that is each S_i is obtained from the true signal C by additive Gaussian noise.*

Problem 2. *In this setup, we assume each $S_i \in \mathcal{S}$ is of the form $S_i = \tilde{C} + \mathbb{M}_{n\times n}(\mathcal{N}(0, \sigma_i))$ where $\tilde{C} = C + \mathbb{M}_{n\times n}(\mathcal{N}(0, \sigma_{input}))$ is constant across S_i. In particular, each S_i is obtained from an already noisy signal $\tilde{C}$ by additive Gaussian noise.*

Unless otherwise stated, we will be using Problem 1.

3.2 Proposed Pipeline and Model Architecture

In this section, we describe our pipeline and model architecture. Each object has several images associated with it:

- The original (clean) image.
- The boundary extraction result of the clean image, denoted C above. We will refer to this as the *clean boundary image*.
- Several noisy images, obtained by adding Gaussian noises to the clean image, denoted $\mathcal{I}$ above.
- The boundary extraction results of all the noisy images, denoted $\mathcal{S}$ above. We will refer to them as the *noisy boundary images*.

As noted in section 3.1, we partition each image into patches to construct our set of batches $\mathcal{B}$ where each batch $B_{(i,j)} \in \mathcal{B}$ corresponds to the set of all $(i, j)^{th}$ patches from the noisy boundary images, or $\mathcal{S}$. Such set of batches $\mathcal{B}$ serves as input to our model, as seen in figure 2.

The architecture of our model is similar to that of a transformer encoder [VSP+17]. It starts with a linear layer, followed by several transformer blocks, and ends with a final linear projection layer. The first linear layer is an embedding layer that projects every input patch in $\mathcal{B}$ into a higher-dimensional space $\mathbb{R}^d$.

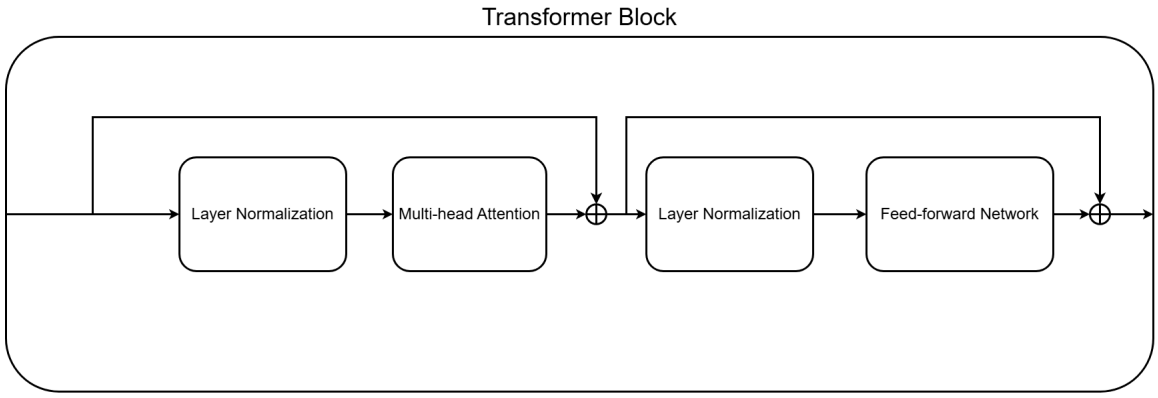

Figure 3: Close up of a single Transformer block within our model in figure 2.

Each of the transformer blocks consists of the multi-head self attention module, followed by a 2-layer feed-forward network with ReLU activation. Similar to the decoder-only transformer architecture used in language models nowadays [RWC+19], we apply layer normalization before the attention or feed-forward module, as shown in figure 3. More precisely, if $X \in \mathbb{R}^{M \times N \times d}$ denotes the input vectors to a transformer block, then we have:

$$\begin{aligned} X' &= \text{LayerNorm}(X) \\ Y &= X + \text{MultiHeadSelfAttention}(X') \\ Y' &= \text{LayerNorm}(Y) \\ Z &= Y + \text{FeedForwardNetwork}(Y') \end{aligned}$$

where Z is the output of the transformer block. The final linear projection layer is applied to the patch embedding at the last position in each sequence to produce the reconstruction of that patch. Each patch reconstruction has the same dimension as any patch in $\mathcal{B}$, and it is compared with the corresponding patch in the clean boundary image to compute the mean-squared error (MSE) loss, which we sum over all patches.

When stitching together overlaying patches ($\tau > 0$) we take the average pixel value over each patch the pixel appears in. Furthermore, if $\tau > 0$ when computing the loss, we apply MSE on the reconstructed image itself, not on a patch-by-patch basis.

3.3 Parameters

Our model has several hyper-parameters. Throughout the paper, we use a model that has 6 transformer blocks and 4 attention heads. The model has an embedding dimension of 768, and we use dropout with $p = 0.1$ to regularize training.

We will explore how changes in patch sizes and stride (τ) affect the model's output in section 5. We further explore how training on multiple noise levels as opposed to a single noise level affects the output.

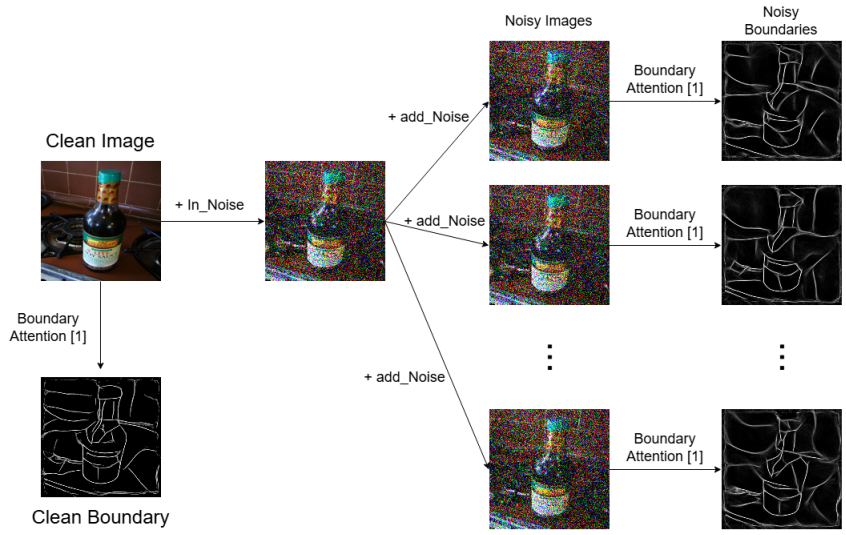

Figure 4: Flow chart of our data creation pipeline for a single image using an example from the Caltech Home Objects 2006 dataset [MP22].

4 Training

We utilize the Intel Image Classification dataset [Ban19] and the Caltech Home Objects 2006 dataset [MP22]. As in figure 4 we iterate over each photo within the union of the two datasets, and for each image we first compute the clean boundary denoted by C in section 3.1. We now proceed in two directions depending on the model and problem setup. For Problem 1, In_Noise in figure 4 would simply be set to $0 \in \mathbb{M}_{n\times n}(\mathbb{R})$. For Problem 2, we have $In\_Noise \sim \mathbb{M}_{n\times n}(\mathcal{N}(0, \sigma_{input}))$. Then we proceed by adding additional noise $add\_Noise \sim \mathbb{M}_{n\times n}(\mathcal{N}(0, \sigma_i))$ (note that σ_i need not be constant across i) to our (potentially noisy) image independently N times to produce a set of noisy images, as shown in figure 4. Finally, we apply $\mathcal{BA}$ (the boundary attention model [PHH+24]) to each noisy image to produce our set of noisy boundaries, denoted $\mathcal{S}$ in section 3.1.

We divide the data into training ($\approx 80\%$), validation ($\approx 15\%$), and test ($\approx 5\%$) datasets. After every epoch of training is completed, we validate the model on the validation dataset and save the checkpoint that has achieved the best validation loss so far. After training, the best checkpoint is evaluated on the test dataset. We use the AdamW optimizer [LH19] throughout training.

5 Experimental Results

In this section, we show some experimental results. First, we consider the case where only one noise level is used to generate the noisy boundary results. In particular, we use Gaussian noises with standard deviation $\sigma = 0.3$. Thus, the model is only trained on boundary

results obtained from noisy images with the same noise level. When performing cross-patch attention, we have the choice of using patches that overlap with each other or patches that do not overlap. We train a model with each option and evaluate its performance. In what follows, we use $\mathcal{A}$ to denote the model trained using non-overlapping patches and $\mathcal{B}$ to denote the model trained using overlapping patches. Figure 5 shows some example images, including the original (clean) image, some samples of the noisy boundary results, the clean boundary image, and the model outputs for both $\mathcal{A}$ and $\mathcal{B}$.

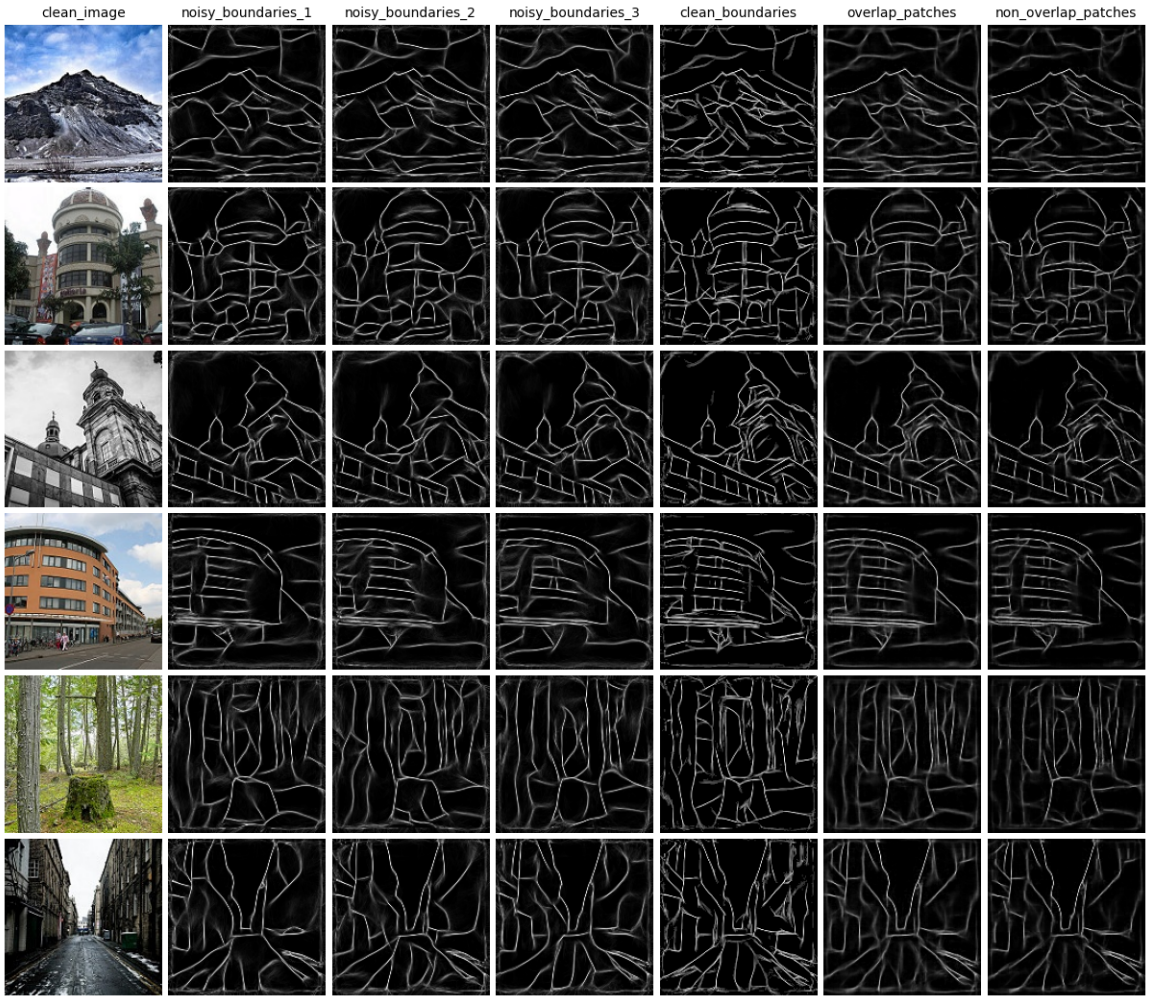

Figure 5: Samples of noisy boundary images, followed by the clean boundary image, and finally the model outputs for $\mathcal{A}$ and $\mathcal{B}$.

Qualitatively, we can see that our model is able to construct some true boundaries that are absent in many (in some cases even all) noisy boundary images (especially in the third and fourth rows). It is also able to remove some noise artifacts. We do not observe a large qualitative difference between the outputs of $\mathcal{A}$ and $\mathcal{B}$, although in some examples (such as the second row), model $\mathcal{A}$ seems to be more confident in certain fine-grained details.

PSNR (Average)	Model $\mathcal{A}$ (Non-overlapping Patches)	Model $\mathcal{B}$ (Overlapping Patches)
Input	15.51	15.51
Model Output	17.68	17.62
Percent Increase	14.00%	13.60%

Table 1: PSNR Values for the boundary detection models taken as an average over the entire test dataset (only 0.3 standard deviation noise level only). Model $\mathcal{A}$ uses non-overlapping patches ($\tau = 0$) of size 10×10 whereas, Model $\mathcal{B}$ uses overlapping patches with stride $\tau = 10$ of size 15×15, all other parameters are constant across the models.

Next, we consider the case where multiple noise levels are used to generate the noisy boundary results. In particular, we use three different standard deviations $\sigma = 0.3, 0.4, 0.5$ to sample Gaussian noises. Due to limitations on computational resources, only a subset of our training data is generated using all three noise levels. Here, we also have the choice of using overlapping or non-overlapping patches to train the model. In what follows, we use $\mathcal{A}'$ to denote the model trained on boundary results with multiple noise levels and non-overlapping patches. $\mathcal{B}'$ denotes the model trained on boundary results with multiple noise levels and overlapping patches. We show some example images in figure 6. On these examples, we observe similar qualitative trends as the one-noise level case.

PSNR (Average)	Model $\mathcal{A}'$ (Non-overlapping Patches)	Model $\mathcal{B}'$ (Overlapping Patches)
Input	15.37	15.37
Model Output	17.57	17.72
Percent Increase	14.31%	15.29%

Table 2: PSNR Values for the boundary detection models taken as an average over the entire test dataset. Model $\mathcal{A}$' uses non-overlapping patches ($\tau = 0$) of size 10×10 whereas, Model $\mathcal{B}$' uses overlapping patches with stride $\tau = 10$ of size 15×15, all other parameters are constant across the models.

We now look at how two of our models perform on Problem 2. We introduce a new model $\mathcal{C}$ trained with the same parameters as $\mathcal{B}'$ (overlapping). However, it is trained on a smaller dataset ($\approx 4,500$ Objects) constructed for Problem 2, the setup where the input image already has some Gaussian noise (we use a standard deviation of 0.2). We add additional noise at standard deviations $\sigma_1 = 0.1$ and $\sigma_2 = 0.15$ (20 times per noise level) to the input image to produce the batch of noisy images to which we apply $\mathcal{BA}$. Note that we also pass in the input boundaries as in figure 7 to our model.

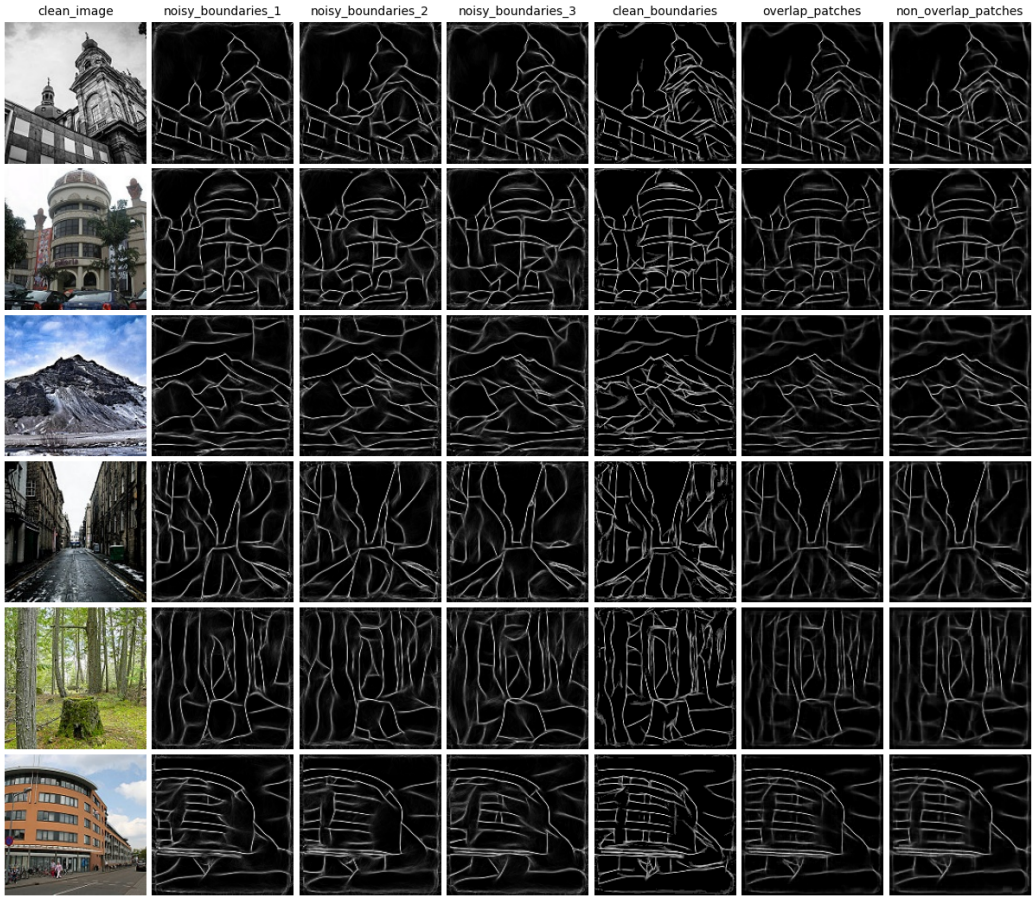

Figure 6: Samples of noisy boundary images, followed by the clean boundary image, and finally the model outputs for $\mathcal{A}'$ and $\mathcal{B}'$.

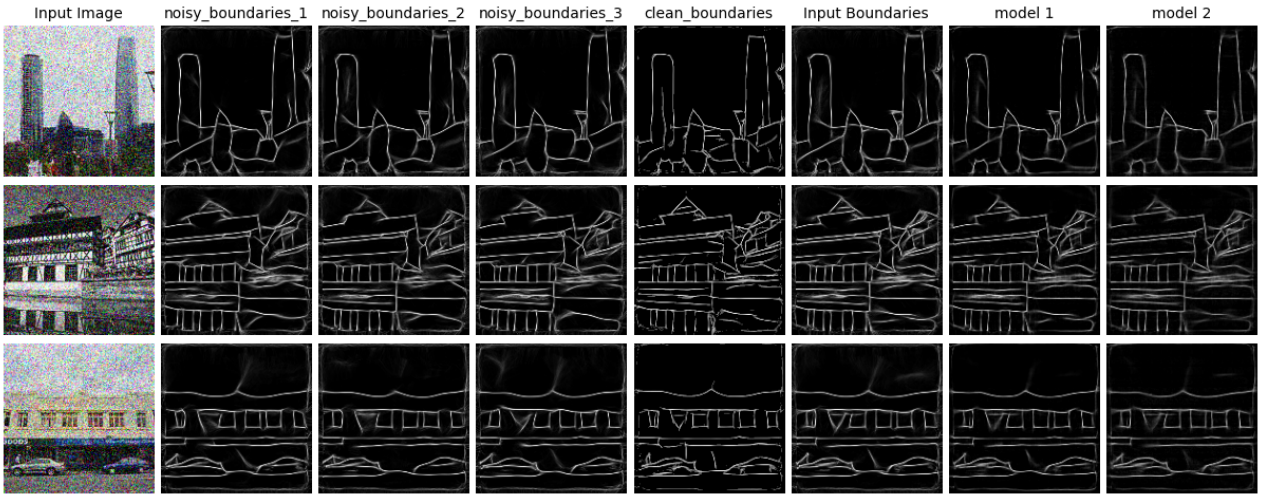

Figure 7: Samples of noisy boundary images obtained from the input (noisy) image by adding more Gaussian noise, followed by the clean (true) boundaries, the input boundary image, and finally the model outputs for $\mathcal{B}'$ (from above, same as model 1 in the figure) and $\mathcal{C}$ which is a model specially trained on a smaller dataset for Problem 2.

PSNR (Average)	Model $\mathcal{B}'$	Model $\mathcal{C}$
Input	16.11	16.11
Model Output	17.30	17.53
Percent Increase	7.39%	8.81%

Table 3: PSNR Values for the boundary detection models taken as an average over the entire test dataset built for Problem 2. Model $\mathcal{B}$' uses overlapping patches with stride $\tau = 10$ of size 15×15 and was trained on the original dataset for Problem 1 above. Model $\mathcal{C}$ is the model specially trained on a smaller dataset for Problem 2 as described above.

We see that both models increase the PSNR; however, model $\mathcal{C}$, the model trained for this purpose, does outperform the previous model $\mathcal{B}'$ as expected, although not by much.

6 Applications

In this section, we analyze how our methodology can be simply translated over to the context of image denoising. In particular, our noisy signals $\mathcal{S}$ are now images with added (gaussian) noise instead of their boundary results. We utilize the same model architecture as in section 3.2, on this new set of inputs.

For our model in this section, we used a patch size of 15×15 pixels with a stride of 10 pixels. We trained our model on added Gaussian noise with standard deviations $0.3, 0.4$, and

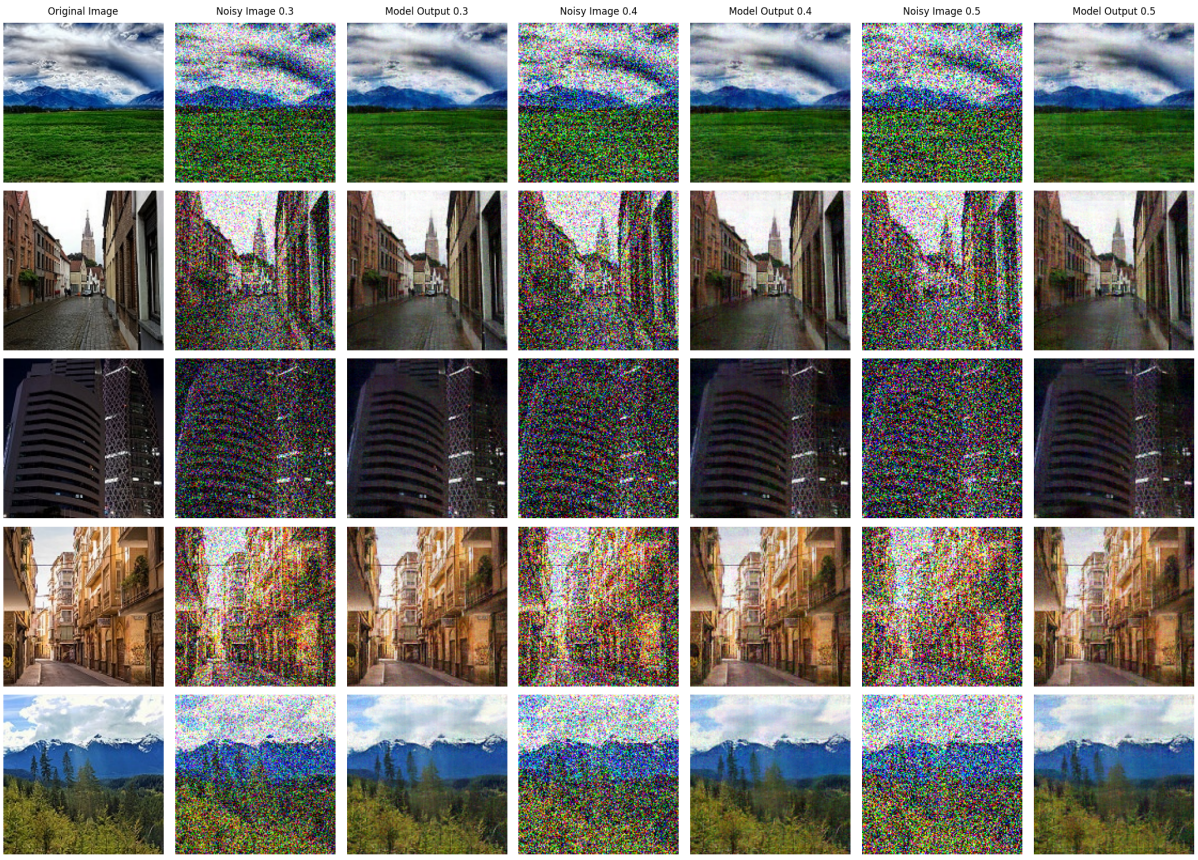

Figure 8: Sample outputs from our model taken from the test dataset ($\approx 5\%$ of all the data). Where the normalized noisy images have additive Gaussian noise with standard deviations $0.3, 0.4$, and 0.5 respectively. Images taken from the Intel Image Classification dataset [Ban19]

0.5 with each noise level consisting of 30 pictures, all other parameters are the same as in section 3.3.

As seen in table 4 our model significantly increases the Peak-Signal-to-Noise-Ratio (PSNR) even at high noise levels. Moreover, looking at figure 8 we see strong results even on the low light image, the image in the third row.

We see a significant increase in PSNR over all noise levels; moreover, we see that the percent increase as depicted in table 5 increases with the noise level (standard deviation). This highlights the robustness of this method to various levels of noise.

7 Discussion

In this section, we discuss our results and propose future directions of research within the context of our method.

PSNR	Image 1	Image 2	Image 3	Image 4	Image 5
Average Input (Noise level 0.3)	12.25	12.28	12.79	12.05	12.28
Output (Noise level 0.3)	26.34	25.98	26.42	25.03	25.75
Average Input (Noise level 0.4)	10.42	10.45	10.75	10.30	10.44
Output (Noise level 0.4)	24.96	24.50	24.95	23.64	24.33
Average Input (Noise level 0.5)	9.21	9.21	9.36	9.16	9.22
Output (Noise level 0.5)	23.91	23.28	23.85	22.49	23.19

Table 4: PSNR Values for the images in figure 8, where Image i corresponds the the image in the i^{th} row. Average Input denotes the average PSNR over all the noisy inputs to the model for that image compared with the Original Image. Output corresponds to the PSNR of our models' output compared to the Original Image.

PSNR (Average)	Noise level 0.3	Noise level 0.4	Noise level 0.5
Input	12.18	10.37	9.19
Model Output	25.82	24.41	23.26
Percent Increase	111.99%	135.39%	153.10%

Table 5: PSNR Values for the denoising model at various noise levels taken as an average over the entire test dataset.

Question 1. *We propose a possible modification of our model which may further improve results. Our model learns on a patch-by-patch basis, but this does not account for the global information of an image, which exists between patches and may help with prediction. Thus, we conjecture that instead of a final linear projection layer, using a more complex decoder model that learns the global information of images might induce better performance. In this way, both local and global information would be utilized to construct the final result.*

Furthermore, in the case of overlapping patches, the case when $\tau > 0$ in section 3.1, the decoder would also be able to stitch the patches together in a learnable manner as opposed to the simple pixel-by-pixel averaging that we employed.

Question 2. *As seen in section 6, our method has interesting applications within the field of image denoising, despite our model not being tailored to such a problem. We hypothesize that our methodology can be used to further improve image denoising results when combined with a more complex decoder model, as detailed in Question 1. We further wonder if our approach can be used in conjunction with an existing denoising model to improve their robustness to variations in noise, in a similar manner to section 3.1 where we improve results from the boundary attention model [PHH+24] on various levels of noise.*

Question 3. *How does our method perform on low-light images; in particular, can such a method be used for boundary detection in low-light scenes?*

Question 4. *As seen in section 5 tables 1 and 3 we see that training over multiple noise levels increases the performance we conjecture that training over randomly sampled noise levels could produce a model which performs more uniformly across noise levels.*

Question 5. *Our model can take in images of various resolutions. It would be interesting to see how it performs on images with higher levels of resolution than the one it was trained on.*

Question 6. *In section 6, we adapted our method to image denoising using the setup of Problem 1. A natural question is to ask whether our method can be adapted for image denoising using the setup of Problem 2. That is, if one is given an already noisy signal, is it possible to denoise it with our method by first adding small amount of additional noises to it to produce many noisy samples?*

8 Conclusion

In conclusion, we have introduced a method for improving the boundary detection results of low SNR images. We have empirically evaluated our method and shown that it can be easily adapted to perform image denoising at a variety of noise levels with a single model. We have also discussed several ideas for future research directions.

References

[Ban19] Puneet Bansal. Intel image classification. `https://www.kaggle.com/puneet6060/intel-image-classification.`, 2019.

[Can86] John Canny. A computational approach to edge detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-8(6):679–698, 1986.

[DBK+21] 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, 2021.

[LH19] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization, 2019.

[MP22] Pierre Moreels and Pietro Perona. Caltech home objects 2006, Apr 2022.

[Ofi24] Nati Ofir. Efficient bayesian detection of faint curved edges in noisy images, 2024.

[OGNB16] Nati Ofir, Meirav Galun, Boaz Nadler, and Ronen Basri. Fast detection of curved edges at low snr, 2016.

[PHH+24] Mia Gaia Polansky, Charles Herrmann, Junhwa Hur, Deqing Sun, Dor Verbin, and Todd Zickler. Boundary attention: Learning curves, corners, junctions and grouping, 2024.

[RBS00] Mark Robertson, Sean Borman, and Robert Stevenson. Estimation-theoretic approach to dynamic range enhancement using multiple exposures. *Journal of Electronic Imaging*, 12, 06 2000.

[RWC+19] Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. 2019.

[SSD20] Farhana Sultana, Abu Sufian, and Paramartha Dutta. Evolution of image segmentation using deep convolutional neural network: A survey. *Knowledge-Based Systems*, 201–202:106062, August 2020.

[VSP+17] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need, 2017.

[VZ21] Dor Verbin and Todd Zickler. Field of junctions: Extracting boundary structure at low snr, Oct 2021.

[XLG24] Wei Xu, Junjie Luo, and Qi Guo. Ct-bound: Robust boundary detection from noisy images via hybrid convolution and transformer neural networks, 2024.