Text Sharpener: Deblurring Text Images with CNN

Tong Xie, Eric Xia, Fiona Wu, Eliza Jiang

Abstract

Blurry text in images poses significant challenges for readability and information extraction, often arising from noisy images, motion blur, poor lighting conditions, etc. In this project, we utilize Convolutional Neural Network (CNN) to denoise and sharpen text images to improve their visual clarity. We extend the U-Net architecture and train the model on synthetic blurry images generated from blur kernels, optimizing over a combined objective of reconstruction and edge loss. This approach significantly improves the readability of images, improving the average Peak signal-to-noise ratio (PSNR) on testing set by 62% from 16.97 to 27.21 dB.

1 Introduction

Image blurring affects readability and, especially, textual information extraction of the original picture. The blurriness can be caused by sudden motion when taking a picture, lighting in the environment, and filters applied directly to the pictures. Many deblurring methods and algorithms have been attempted to capture clear images from the blurry versions, including capturing hand movements for clinical diagnosis and detecting traffic light status for autonomous vehicles [9][8].

In this project, we focused on textbook pictures with upright, horizontal words that were blurred with manually added filters. Our goal was to deblur those pictures so that the words could be recognized. Pictures with text are often hard to read, even if just a few noises exist. The results of our model aim to sharpen the blurry text.

Traditionally, blurred and unclear images can be sharpened by adjusting image filters and Support Vector Machine (SVM), which are time-consuming and computationally expensive since they are often applied case by case [4][2]. In this work, we utilized the CNN architecture, which is particularly effective for pattern recognition, classification, and other tasks that involve learning from spatial hierarchies. We extended the UNet design to adapt specifically for the task of deblurring images with text, showing great sharpening effects on blurred textbook images with minimal human labor.

2 Background

Convolutional Neural Network (CNN) is an architecture in deep neural networks designed to process data with grid-like topology [5]. It is particularly effective for pattern recognition, classification, and other tasks that involve learning from spatial hierarchies. CNN processes the input data I via forward propagation, where I undergoes successive transformations:

$$H^{(l+1)} = h(W^{(l)} * H^{(l)} + b^{(l)}), \tag{1}$$

where $H^{(l)}$ is input to the l-th layer, $W^{(l)}$ and $b^{(l)}$ are the learnable weights and biases, and h is the activation function.

Specifically, CNN typically consists of convolutional layers, pooling layers, activation functions, and fully connected layers, which collectively extract, process, and classify hierarchical features from input data. Mathematically, the convolution operation is defined as:

$$f[i,j] = (I * K)[i,j] = \sum_m \sum_n I[i+m, j+n] \cdot K[m,n], \tag{2}$$

where $f[i,j]$ is the ij-th entry of the feature map, I is the input image, K is the learnable kernel, and m, n are coordinates in kernel. This preserves spatial relationships in data and translational invariance. Pooling layers then reduce the dimensions of feature maps to retain essential features. For example, max pooling captures the most pronounced feature in each patch of feature map values fro a window size of $n \times n$,

$$P_{\max}(i,j) = \max_{a=0}^{n-1} \max_{b=0}^{n-1} f(i \cdot s + a, j \cdot s + b), \tag{3}$$

where s is the stride of the pooling window, and a, b iterate over the window dimensions. Activation functions such as ReLU [1] introduce non-linearity within CNN for complex mapping between inputs and outputs. While CNN is common for pattern extraction, it can also be extended for other tasks. For example, the fully connected layers that follow flatten the reduced feature maps and allow for objectives such as classification, where the final layer produces probabilistic outputs using the softmax given by

$$P(y = k|x) = \frac{\exp(z_k)}{\sum_j \exp(z_j)}. \tag{4}$$

3 Dataset

To train and evaluate our deblurring model, we curated a custom dataset specifically focused on textual images.

Data Collection. The dataset was sourced from the textbook *Pattern Recognition and Machine Learning (2006)* by Christopher Bishop [3] with the following steps:

- **Image Cropping:** Each page of the textbook was processed to extract 5 random crops of size 128×128 pixels.
- **Text Filtering:** To ensure all samples contained sufficient text, we used EasyOCR, an optical character recognition tool, to identify and count the number of words in each crop. Only crops with more than 10 words were kept.
- **Clear Dataset:** After filtering, we obtained a dataset consisting of 1403 grayscale images, each having size 128×128 pixels, as the targets for the deblurring model.

Blurring Process. To simulate real-world blurry images, we applied motion blur to each clean image. We used convolutional kernels to replicate the effects of motion blur, where the blur kernels are as follows:

- **Kernel Size:** We used blur kernels of varying sizes (3x3, 6x6, or 9x9) to create blurs of different intensities. Larger kernel sizes produce more pronounced blurring effects.
- **Blur Angle:** The blur direction was varied by rotating the kernel to random angles between 0° and 180° in increments of 15°. This simulates motion blur occurring in different orientations.

Each image is blurred using a randomly selected kernel size and blur angle, ensuring variation in the blurring effect. The resulting images are used as model inputs, as shown in Figure 1 (a).

Normalization. Before feeding the data into the model, pixel intensities were normalized to the range [0, 1] by dividing each pixel value by 255.
Dataset Splits. The dataset was divided into three subsets for training, validation, and testing to ensure there is no data leakage:

- **Test Set:** A fixed set of 140 images (10%) was reserved to evaluate model performance.
- **Training and Validation Sets:** The remaining 1263 images were randomly split into 80% training (1010 images) and 20% validation (253 images).

Ethical Concerns. This project is entirely for educational purposes, with textbook materials accessed directly through the official Microsoft website.

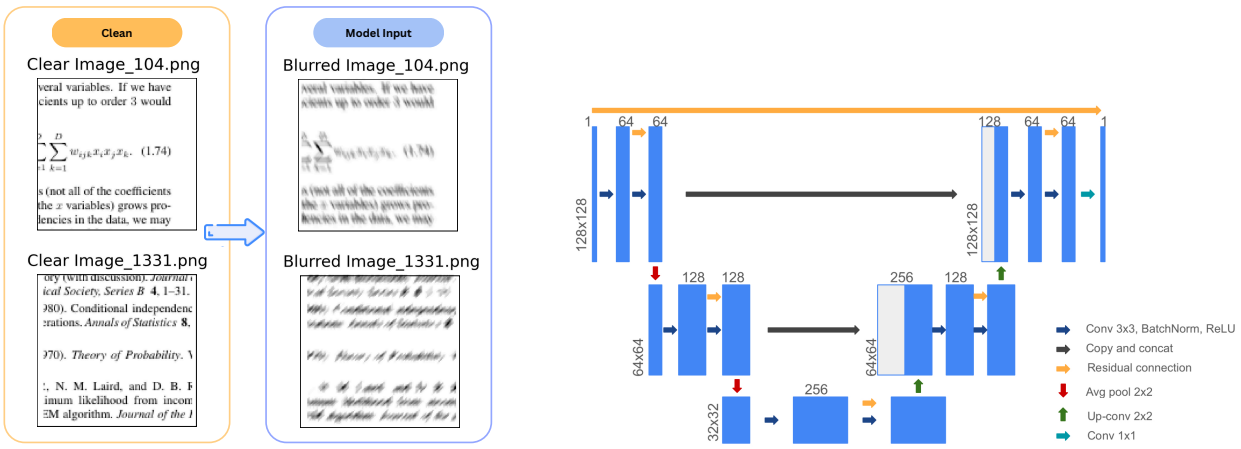

Figure 1: (a) Dataset sample visualization and (b) UNetV2 model architecture.

4 Model

UNetV2 extends the model design proposed in *U-Net: Convolutional Networks for Biomedical Image Segmentation* [7]. Several modifications were made to adapt for grayscale image deblurring. A visualization is shown in Figure 1 (b) and we detail the architecture below:

- **Input and Output Dimensions:** The model takes as input 128×128 grayscale images with a single channel and outputs deblurred images of the same dimensions and channel.
- **Double Residual Convolution Blocks:** Each convolutional block in the encoder and decoder path comprises two convolutional layers (3×3 kernel, stride 1, padding 1) followed by Batch Normalization and ReLU activation. A residual connection[6] is added within each block, enabling better gradient flow during training.
- **Downsampling and Bottleneck:** The encoder path consists of two downsampling stages. Each stage reduces the spatial dimensions by half using a 2×2 average pooling layer while doubling the feature dimensions:
  - 128×128 → 64×64 (64 features)
  - 64×64 → 32×32 (128 features)

  The bottleneck layer at the smallest spatial resolution (32×32) doubles the feature depth to 256.

- **Upsampling Path with Skip Connections:** The decoder path mirrors the encoder path and progressively upsamples the feature maps to reconstruct the original resolution. Each upsampling operation uses a 2×2 transposed convolution to increase spatial dimensions:
  - 32x32 → 64×64 (128 features)
  - 64×64 → 128x128 (64 features)

  Skip connections copy features from the encoder and concatenate them with the upsampled features in the decoder, before passing through more convolution blocks.
- **Final Convolution and Direct Input-to-Output Connection:** A 1×1 convolution at the final layer reduces the feature depth back to a single channel to match the grayscale output. Unlike the encoder-decoder skip connections, the final skip connection directly adds the input to the model's output. This helps the network focus on learning residual deblurring functions.

Modifications to the original UNet:

- **Residual connections** added to double convolution blocks for better gradient flow.
- **Average pooling** used instead of max pooling in downsampling layers to retain context.
- **Padding** applied to convolutional layers to prevent spatial size reduction.
- **Global skip connection** directly from input to output for residual learning.

5 Methodology

5.1 Objective Function

For the task of denoising blurred images with text, our objective function combines two complementary terms: (1). a **reconstruction loss** and (2). en **edge loss**. This emphasizes edge preservation while reconstructing the clean image, enhancing text clarity and sharpness.

Reconstruction Loss. This term ensures a close match between the denoised image and the clean ground truth. A general form is defined as

$$\mathcal{L}_{\text{reconstruct}} = \frac{1}{N} \sum_{i=1}^{N} \ell \left(I_{\text{target}}^{(i)}, I_{\text{pred}}^{(i)} \right), \tag{5}$$

where $\ell(\cdot)$ denotes a distance function such as $\mathcal{L}_1$ or $\mathcal{L}_2$ loss.

Edge Loss. To emphasize the preservation of sharp edges, we define an edge loss using the Sobel operator. This extracts edge information by computing intensity gradient in the horizontal and vertical directions. For an image I, the gradients are given by

$$\mathbf{G}_x = I * \mathbf{S}_x, \quad \mathbf{G}_y = I * \mathbf{S}_y, \tag{6}$$

where $\mathbf{S}_x$ and $\mathbf{S}_y$ are the Sobel kernels:

$$\mathbf{S}_x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}, \quad \mathbf{S}_y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}.$$

The edge magnitude $\mathbf{G}$ is defined as:

$$\mathbf{G} = \sqrt{\mathbf{G}_x^2 + \mathbf{G}_y^2 + \epsilon}, \tag{7}$$

where $\epsilon > 0$ is a small constant for numerical stability. For a prediction $I^{(i)}_{\text{pred}}$ and target $I^{(i)}_{\text{target}}$, the edge loss is given by the following with $\mathcal{L}_2$ distance:

$$\mathcal{L}_{\text{edge}} = \frac{1}{N} \sum_{j=1}^{J} \left\| \mathbf{G}^{(j)}_{\text{pred}} - \mathbf{G}^{(j)}_{\text{target}} \right\|^2, \tag{8}$$

where $\mathbf{G}_{\text{pred}}$, $\mathbf{G}_{\text{target}}$ are edge magnitude maps of the predicted and target images respectively, i iterates over all pixel locations in image, and J is the total number of pixels (e.g. for an RGB image, $J = \text{height} \times \text{width} \times 3$).

Combined Loss. The final objective used in training is a weighted combination of the reconstruction loss and edge loss:

$$\mathcal{L} = \mathcal{L}_{\text{reconstruct}} + \lambda \mathcal{L}_{\text{edge}}, \tag{9}$$

where λ is a hyperparameter controlling the intensity of edge preservation.

5.2 Training

Aspect	**Details**
Optimizer	AdamW, initial learning rate of $\eta = 0.01$, lr reduced on plateau (factor=0.5, patience=60), gradient clipping = 1
Epochs	1000 epochs
Batch Size	128
Loss Function	Combined Loss $= \mathcal{L}_{\text{reconstruct}} + \lambda \mathcal{L}_{\text{edge}}, \mathcal{L}_{\text{reconstruct}} = \mathcal{L}_1, \lambda = 0.3$
Metrics	• Loss: measured from the optimization objective • Peak Signal-to-Noise Ratio (PSNR), calculated as: $\text{PSNR} = 10 \cdot \log_{10} \left(\frac{\text{MAX}^2}{\text{MSE}} \right)$, where MAX is the maximum pixel value (e.g. 255) and MSE is the mean squared error
Evaluation	Evaluated on both training and validation datasets using loss and PSNR. The best model is selected based on the highest PSNR on the validation set.

Algorithm 1 Training Pipeline for CNN Denoiser

Require: Clean dataset $\{I_{\text{clean}}\}$, noise generator Noise$(\cdot)$, model f_θ, loss function $\mathcal{L}(\cdot)$, learning rate η, batch, epochs T.

for epoch $t = 1, \ldots, T$ **do**
  for batch $\{I_{\text{clean}}\}$ **do**
    $I_{\text{noisy}} \leftarrow I_{\text{clean}} + \epsilon, \epsilon \sim \text{Noise}(\cdot)$ *Add noise*
    $I_{\text{pred}} \leftarrow f_\theta(I_{\text{noisy}})$ *Predict clean*
    $\mathcal{L} \leftarrow \mathcal{L}(I_{\text{pred}}, I_{\text{clean}})$ *Compute loss*
    $\theta \leftarrow \theta - \eta \nabla_\theta \mathcal{L}$ *Update parameters*
  end for
end for

6 Results

Two key metrics are used for model performance evaluation and visualized in Figure 2:

- **Peak Signal-to-Noise Ratio (PSNR):** Figure 2 (a) shows improvement in PSNR values over the course of training for both training and validation data sets. As training PSNR and validation PSNR exhibit similar increasing trend and convergence, the training process is effective, and the model generalizes well to validation data.
- **Training/validation loss over epochs:** Figure 2 (b) illustrates the training and validation loss curves. The losses decrease over epochs, with the validation loss converging to the training loss, suggesting an optimal fit with balance between overfitting and underfitting.

The average PSNR on the test set is 27.21 dB, achieving substantial improvement from the 16.97 dB of the original set and a 62% improvement in image quality. The model performance is visualized in Figure 3, showing the success of the UNetV2 model in deblurring text images.

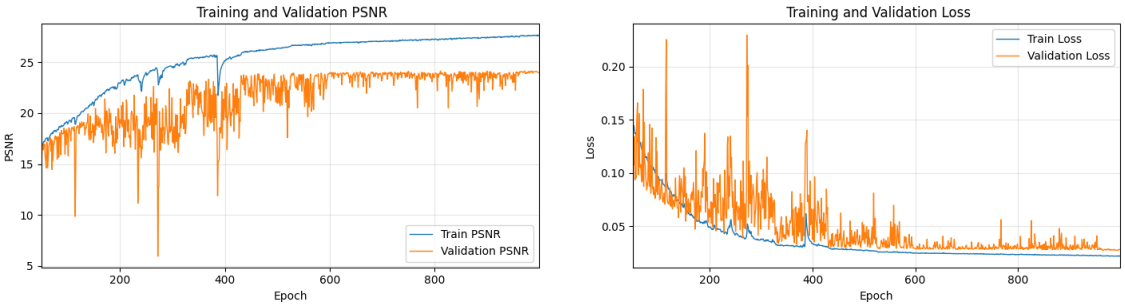

Figure 2: (a) UNetV2 Training and Validation PSNR and (b) UNetV2 Training and Validation Loss.

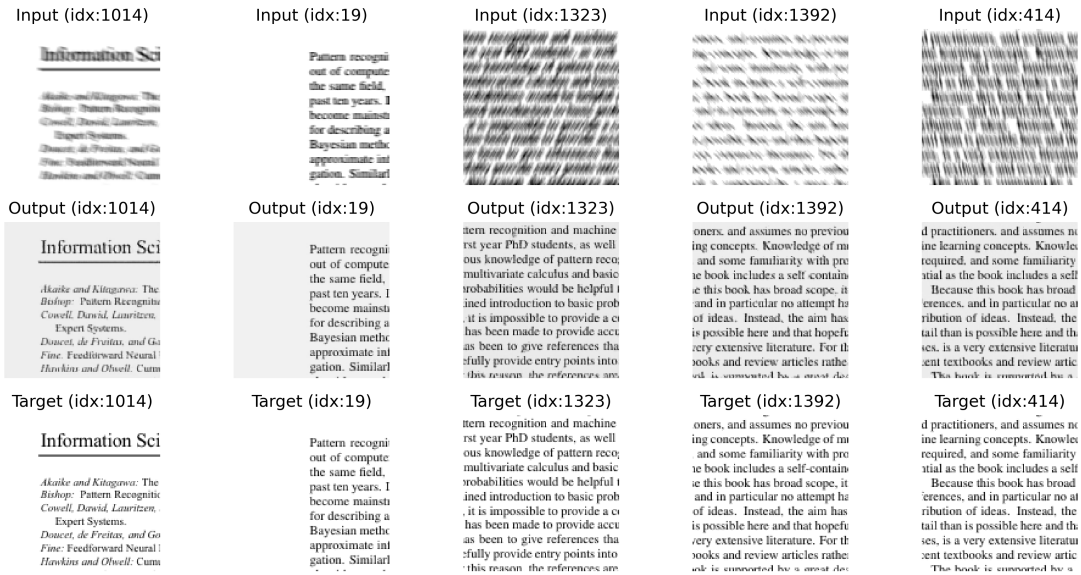

Figure 3: Sample Model Output Visualization.

7 Conclusion

Sharpening blurry text presents many challenges, especially when applied to highly variable images containing multiple types of blur. Initially, our goal was to develop a model capable of handling various types of blurred images, including noise and motion blur. However, we quickly realized that this approach led to generalizing difficulties as the input data was difficult to learn. Recognizing this limitation, we pivoted to a more focused dataset of text data exclusively from textbooks. This switch allowed for improving model performance and interpretability.

The results of both models demonstrated significant improvements in image quality. Based on the PSNR evaluation metric, both models enhanced the reconstructed images to resemble the ground truths. The results are also promising on the basis of visual comparisons. By comparing the two models, UNetV2 performs better with 27.21 db PSNR. However, limitations persists:
Limitations:

- *Lack of generalization ability:* When applied to data outside the textbook dataset (real-world blurred images), the model shows reduced performance.
- *Sample inefficiency:* More variable data is required to perform extensive training for optimization purposes.

Potential Improvements:

- Conduct a comprehensive performance comparison across models and modifications.
- Perform more extensive hyperparameter tuning.
- Expand the dataset with varied blur kernels and additional noise types like Gaussian.
- Evaluate model generalization on real-world text images.
- Scale up the dataset and model to handle general textual image denoising and deblurring.

8 Additional Components

Code: `https://github.com/Ruichu-Xia/text-sharpener`
Author contributions: Tong tested the baseline models and designed the loss functions along with the corresponding training pipeline. Eric collected the dataset and built the best-performing version of UNet model presented in the report. Fiona explored the UNet model and modified the structure of UNet adapting to the context of text deblurring. Eliza explored on UNet and BasicNet models, along with various loss functions and hyperparameters for ablation studies.
Acknowledgments: We extend our gratitude to Professor Lara Kassab for her insightful advice and guidance throughout the course of Math 156.
Checklist of Requirements: All requirements are included in our project.

References

[1] Abien Fred Agarap. Deep learning using rectified linear units (relu), 2019.

[2] Susan Alston and Alexander Cook. A review of the development and application of the pressurized residual method (press) for calibration and validation in analytical chemistry. *Chemometrics and Intelligent Laboratory Systems*, 177:63–74, 2018.

[3] Christopher M Bishop and Nasser M Nasrabadi. *Pattern recognition and machine learning*, volume 4. Springer, 2006.

[4] Nidhal El abbadi. improve image de-bluring. 04 2018.

[5] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. `http://www.deeplearningbook.org`.

[6] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016.

[7] Thomas Brox Olaf Ronneberger, Philipp Fischer. U-net: Convolutional networks for biomedical image segmentation. *arXiv preprint arXiv:1505.04597*, 2015.

[8] Sanjay Saini, S Nikhil, Krishna reddy Konda, Harish Bharadwaj, and N Ganeshan. An efficient vision-based traffic light detection and state recognition for autonomous vehicles. pages 606–611, 06 2017.

[9] Wei Zhang, Ming Li, and Xiaohong Chen. A novel approach to data analysis using machine learning techniques. *Expert Systems with Applications*, 207:117921, 2023.