SUPER RESOLUTION IMAGE DENOISER NETWORK: TOWARD REAL LIFE IMAGE NOISE REMOVAL

P. MOHIT HARSH - 21951A04A3

SUPER RESOLUTION IMAGE DENOISER NETWORK: TOWARD REAL LIFE IMAGE NOISE REMOVAL

A Project Report
submitted in partial fulfilment of the
requirements for the award of the degree of

Bachelor of Technology

in

Electronics and Communication Engineering

By

P. MOHIT HARSH – 21951A04A3

Under The Guidance Of

Dr V. Padmanabha Reddy

Professor

Department of Electronics and Communication Engineering

INSTITUTE OF AERONAUTICAL ENGINEERING

(Autonomous)

Dundigal, Hyderabad – 500 043, Telangana

November, 2024

DECLARATION

I certify that,

a. The work contained in this report is original and has been done by me under the guidance of my supervisor(s).

b. The work has not been submitted to any other Institute for any degree or diploma.

c. I have followed the guidelines provided by the Institute in preparing the report.

d. I have conformed to the norms and guidelines given in the Ethical Code of Conduct of the Institute.

e. Whenever I have used materials (data, theoretical analysis, figures, and text) from other sources, I have given due credit to them by citing them in the text of the report and giving their details in the references. Further, I have taken permission from the copyright owners of the sources, whenever necessary.

Place: **Signature of Student(s)**

Date: **21951A04A3**

CERTIFICATE

This is to certify that the project report entitled **SUPER RESOLUTION IMAGE DENOISER NETWORK: TOWARD REAL LIFE IMAGE NOISE REMOVAL** submitted by **P. Mohit Harsh** to the Institute of Aeronautical Engineering, Hyderabad in partial fulfillment of the requirements for the award of the Degree Bachelor of Technology in Department of Electronics and Communication Engineering is a bonafide record of work carried out by them under our guidance and supervision. The contents of this report, in full or in parts, have not been submitted to any other Institute for the award of any Degree.

Supervisor	**Head of Department**
Dr V. Padmanabha Reddy	Dr P. Munaswamy
Professor	Professor & Head

Date:

APPROVAL SHEET

This project report entitled **SUPER RESOLUTION IMAGE DENOISER NETWORK: TOWARD REAL LIFE IMAGE NOISE REMOVAL** by **P. Mohit Harsh** is approved for the award of the Degree Bachelor of Technology in the Department of Electronics and Communication Engineering.

Examiner(s)

Supervisor

Dr V. Padmanabha Reddy

Professor, ECE

Principal

Dr. L V Narasimha Prasad

Date:

Place:

ACKNOWLEDGEMENT

I am deeply grateful to my project guide, **Dr. V. Padmanabha Reddy**, from the Department of Electronics and Communication Engineering, for his invaluable guidance, constant support, and inspiration which have sustained me to accomplish my work successfully. Without his valuable inputs and expertise, this project would not have been possible.

I want to express my heartfelt gratitude to **Dr P. Munaswamy**, Head of the Department, for helping me discover my hidden talents, supporting my professional development, boosting my confidence, nurturing my commitment and organization skills, and guiding me towards success.

I would like to take this opportunity to thank Principal, **Dr. L.V. Narasimha Prasad**, and the Management for providing me with the necessary facilities and opportunities that helped me to complete my project work successfully. I would like to extend my gratitude to the teaching and non-teaching faculty members of the Department of Electronics and Communication Engineering for their support, guidance, and the facilities provided during my project work.

I would like to offer my gratitude to my Parents for their unwavering support, encouragement, keen appreciation, and active interest in my academic achievements. Their constant presence and support have been a great source of motivation for me throughout my academic journey. This project has been a significant milestone in my career development, and I intend to use the skills and knowledge I gained in the best possible way. I look forward to continuing my cooperation with all of you in the future.

With Gratitude,

P. Mohit Harsh (21951A04A3)

ABSTRACT

The implementation of Convolution Neural Networks (CNN) for image denoising has been a key field for research in past 10 years. Current state of the art models have been implemented using Transformer and Generative Adversarial Neural Network (GAN) concepts which have high hardware requirements and also require large amounts of data to train. Although they perform very well in controlled lab environment, they are not useful in many real life applications. To tackle the aforementioned problems, I designed a Super Resolution Image Denoiser Network (SRIDNet) that demonstrates a favorable balance between performance and efficiency. It delivers competitive denoising results while maintaining a small model size and low computational cost, making it highly suitable for resource-constrained environments.

Keywords: Convolution Neural Network (CNN), Adversarial Neural Network (GAN), Super Resolution Image Denoiser Network (SRIDNet).

CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

CNN – Convolutional neural network

CBDNet – Convolutional Blind denoising network

SRIDNet – Super Resolution Image Denoising Network

RIDNet - Residual Image Denoising Network

PSNR – Peak Signal to Noise Ratio

SSIM – Structural Similarity Index Measure

CHAPTER-1

INTRODUCTION

1.1 INTRODUCTION

Image denoising is a critical preprocessing step in many computer vision and image processing applications, aimed at restoring corrupted images by reducing noise while preserving important structural details. Traditional denoising techniques, such as Gaussian filtering and wavelet transforms, have long been employed for this task but are often limited by their inability to adapt to complex and diverse noise patterns encountered in real-world images. In recent years, the rise of deep learning has revolutionized the field of image denoising, offering powerful data-driven solutions capable of learning intricate noise distributions and producing visually appealing results.

Convolutional Neural Networks (CNNs), in particular, have become the dominant paradigm for image denoising, leveraging their ability to capture local spatial hierarchies through multiple layers of convolutional filters. These models have significantly outperformed classical approaches, achieving stateof-the-art results by learning from large datasets of noisy and clean image pairs. Recent advancements have also introduced more sophisticated architectures such as Generative Adversarial Networks (GANs) and Transformer models, further improving denoising performance by capturing global dependencies and generating more realistic outputs. Despite their impressive results in controlled environments, these state-of-the-art models often come with high computational costs, extensive memory requirements, and the need for large training datasets. These factors limit their practicality in real-world applications, particularly in resource-constrained environments where hardware and data availability are restricted.

To address these challenges, we propose the Super Resolution Image Denoiser Network (SRIDNet), a novel approach that strikes a balance between denoising performance and computational efficiency. SRIDNet is designed to deliver competitive results with significantly lower hardware demands and reduced model complexity, making it suitable for deployment in practical settings.

1.2 DEEP NEURAL NETWORKS

A deep neural network (DNN) is an ANN with multiple hidden layers between the input and output layers. Similar to shallow ANNs, DNNs can model complex non-linear relationships.

The main purpose of a neural network is to receive a set of inputs, perform progressively complex calculations on them, and give output to solve real world problems like classification. We restrict ourselves to feed forward neural networks.

We have an input, an output, and a flow of sequential data in a deep network. Neural networks are widely used in supervised learning and reinforcement learning problems. These networks are based on a set of layers connected to each other.

In deep learning, the number of hidden layers, mostly non-linear, can be large; say about 1000 layers. DL models produce much better results than normal ML networks. We mostly use the gradient descent method for optimizing the network and minimizing the loss function.
We can use the ImageNet, a repository of millions of digital images to classify a dataset into categories like cats and dogs.

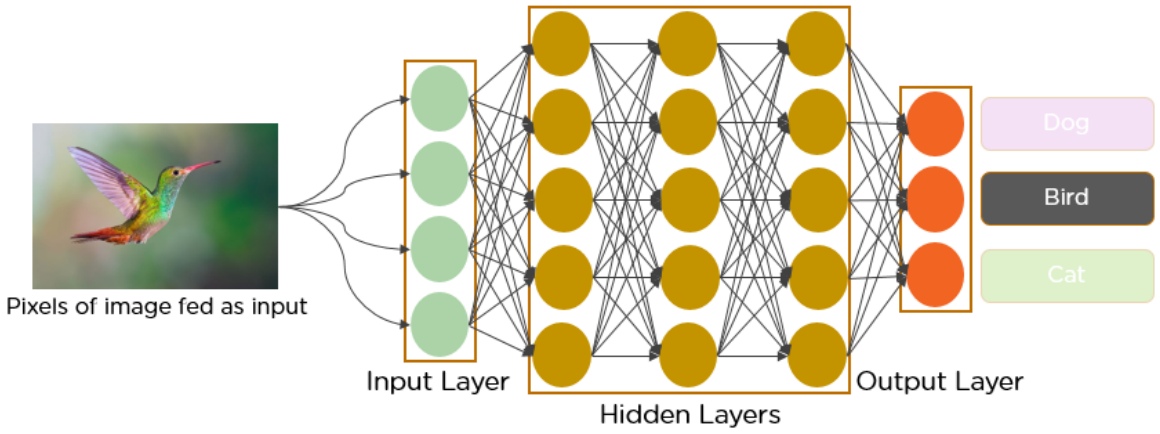

Figure 1. Deep Convolutional Neural Network

DL nets are increasingly used for dynamic images apart from static ones and for time series and text analysis.

Training the data sets forms an important part of Deep Learning models. In addition, Backpropagation is the main algorithm in training DL models. DL deals with training large neural networks with complex input output transformations. One example of DL is the mapping

of a photo to the name of the person(s) in photo as they do on social networks and describing a picture with a phrase is another recent application of DL.

Neural networks are functions that have inputs like x1, x2, x3…that are transformed to outputs like z1, z2, z3 and so on in two (shallow networks) or several intermediate operations also called layers (deep networks). The weights and biases change from layer to layer. ‘w’ and ‘v’ are the weights or synapses of layers of the neural networks.

The best use case of deep learning is the supervised learning problem. Here, we have large set of data inputs with a desired set of outputs. Backpropagation Algorithm. Here, we apply back propagation algorithm to get correct output prediction. The most basic data set of deep learning is the MNIST, a dataset of handwritten digits. We can train deep a Convolutional Neural Network with Keras to classify images of handwritten digits from this dataset. The firing or activation of a neural net classifier produces a score.

For example, to classify patients as sick and healthy, we consider parameters such as height, weight and body temperature, blood pressure etc. A high score means patient is sick and a low score means he is healthy. Each node in output and hidden layers has its own classifiers. The input layer takes inputs and passes on its scores to the next hidden layer for further activation and this goes on till the output is reached. This progress from input to output from left to right in the forward direction is called forward propagation.

Credit assignment path (CAP) in a neural network is the series of transformations starting from the input to the output. CAPs elaborate probable causal connections between the input and the output. CAP depth for a given feed forward neural network or the CAP depth is the number of hidden layers plus one as the output layer is included. For recurrent neural networks, where a signal may propagate through a layer several times, the CAP depth can be potentially limitless.

1.3 CONVOLUTIONAL NEURAL NETWORKS

A Convolutional Neural Network (CNN) is a type of Deep Learning neural network architecture commonly used in Computer Vision. Computer vision is a field of Artificial Intelligence that enables a computer to understand and interpret the image or visual data.
When it comes to Machine Learning, Artificial Neural Networks perform really well. Neural Networks are used in various datasets like images, audio, and text. Different types of Neural Networks are used for different purposes, for example for predicting the sequence of words we use Recurrent Neural Networks more precisely an LSTM, similarly for image classification we use Convolution Neural networks. In this blog, we are going to build a basic building block for CNN.

In a regular Neural Network, there are three types of layers:

1. **Input Layers:** It's the layer in which we give input to our model. The number of neurons in this layer is equal to the total number of features in our data (number of pixels in the case of an image).
2. **Hidden Layer:** The input from the Input layer is then fed into the hidden layer. There can be many hidden layers depending on our model and data size. Each hidden layer can have different numbers of neurons which are generally greater than the number of features. The output from each layer is computed by matrix multiplication of the output of the previous layer with learnable weights of that layer and then by the addition of learnable biases followed by activation function which makes the network nonlinear.
3. **Output Layer:** The output from the hidden layer is then fed into a logistic function like sigmoid or softmax which converts the output of each class into the probability score of each class.

The data is fed into the model and output from each layer is obtained from the above step is called feedforward, we then calculate the error using an error function, some common error functions are cross-entropy, square loss error, etc. The error function measures how well the network is performing. After that, we backpropagate into the model by calculating the derivatives. This step is called Backpropagation which basically is used to minimize the loss.

Convolution Neural Network

Convolutional Neural Network (CNN) is the extended version of artificial neural networks (ANN) which is predominantly used to extract the feature from the grid-like matrix dataset. For example, visual datasets like images or videos where data patterns play an extensive role.

CNN architecture

Convolutional Neural Network consists of multiple layers like the input layer, Convolutional layer, Pooling layer, and fully connected layers.

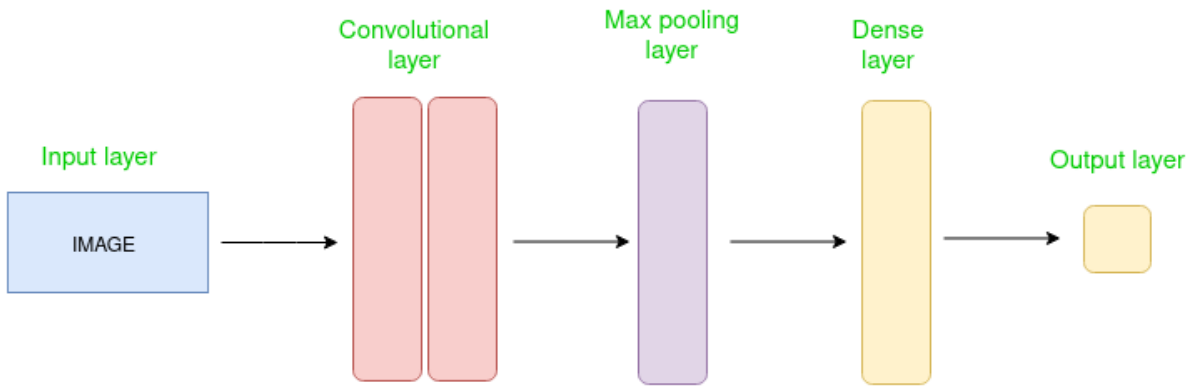

Figure 2. CNN Architecture

The Convolutional layer applies filters to the input image to extract features, the Pooling layer down samples the image to reduce computation, and the fully connected layer makes the final prediction. The network learns the optimal filters through backpropagation and gradient descent.

How Convolutional Layers works

Convolution Neural Networks or covnets are neural networks that share their parameters. Imagine you have an image. It can be represented as a cuboid having its length, width (dimension of the image), and height (i.e. the channel as images generally has red, green, and blue channels).

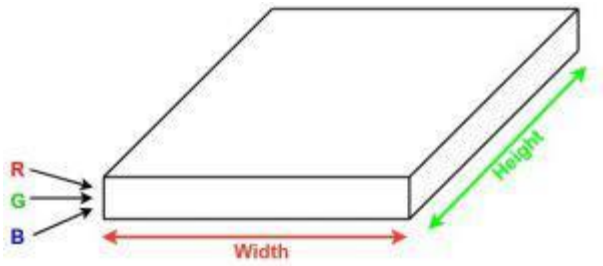

Figure 3. Convolution Layer

Now imagine taking a small patch of this image and running a small neural network, called a filter or kernel on it, with say, K outputs and representing them vertically. Now slide that neural network across the whole image, as a result, we will get another image with different widths, heights, and depths. Instead of just R, G, and B channels now we have more channels but lesser width and height. This operation is called Convolution. If the patch size is the same as that of the image it will be a regular neural network. Because of this small patch, we have fewer weights.

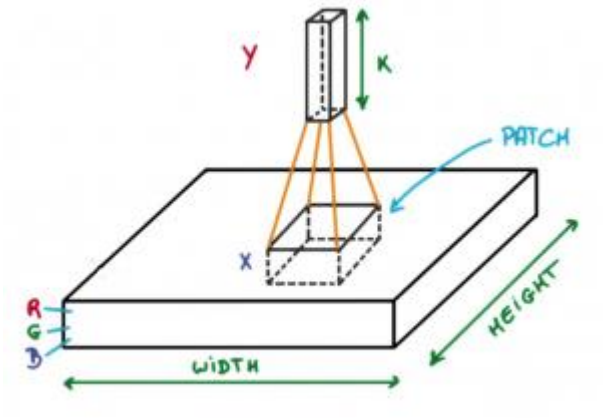

Figure 4. Convolution Patch

Now let's talk about a bit of mathematics that is involved in the whole convolution process.

- Convolution layers consist of a set of learnable filters (or kernels) having small widths and heights and the same depth as that of input volume (3 if the input layer is image input).
- For example, if we have to run convolution on an image with dimensions 34x34x3. The possible size of filters can be axax3, where 'a' can be anything like 3, 5, or 7 but smaller as compared to the image dimension.
- During the forward pass, we slide each filter across the whole input volume step by step where each step is called stride (which can have a value of 2, 3, or even 4 for high-

dimensional images) and compute the dot product between the kernel weights and patch from input volume.

- As we slide our filters, we'll get a 2-D output for each filter and we'll stack them together as a result, we'll get output volume having a depth equal to the number of filters. The network will learn all the filters.

Layers used to build ConvNets

A complete Convolution Neural Networks architecture is also known as covnets. A covnets is a sequence of layers, and every layer transforms one volume to another through a differentiable function.

Let's take an example by running a covnets on of image of dimension 32 x 32 x 3.

- **Input Layers:** It's the layer in which we give input to our model. In CNN, Generally, the input will be an image or a sequence of images. This layer holds the raw input of the image with width 32, height 32, and depth 3.
- **Convolutional Layers:** This is the layer, which is used to extract the feature from the input dataset. It applies a set of learnable filters known as the kernels to the input images. The filters/kernels are smaller matrices usually 2×2, 3×3, or 5×5 shape. it slides over the input image data and computes the dot product between kernel weight and the corresponding input image patch. The output of this layer is referred as feature maps. Suppose we use a total of 12 filters for this layer we'll get an output volume of dimension 32 x 32 x 12.
- **Activation Layer:** By adding an activation function to the output of the preceding layer, activation layers add nonlinearity to the network. it will apply an element-wise activation function to the output of the convolution layer. Some common activation functions are RELU: max(0, x), Tanh, Leaky RELU, etc. The volume remains unchanged hence output volume will have dimensions 32 x 32 x 12.
- **Pooling layer:** This layer is periodically inserted in the covnets, and its main function is to reduce the size of volume which makes the computation fast reduces memory and also prevents overfitting. Two common types of pooling layers are max pooling and average pooling. If we use a max pool with 2 x 2 filters and stride 2, the resultant volume will be of dimension 16x16x12.

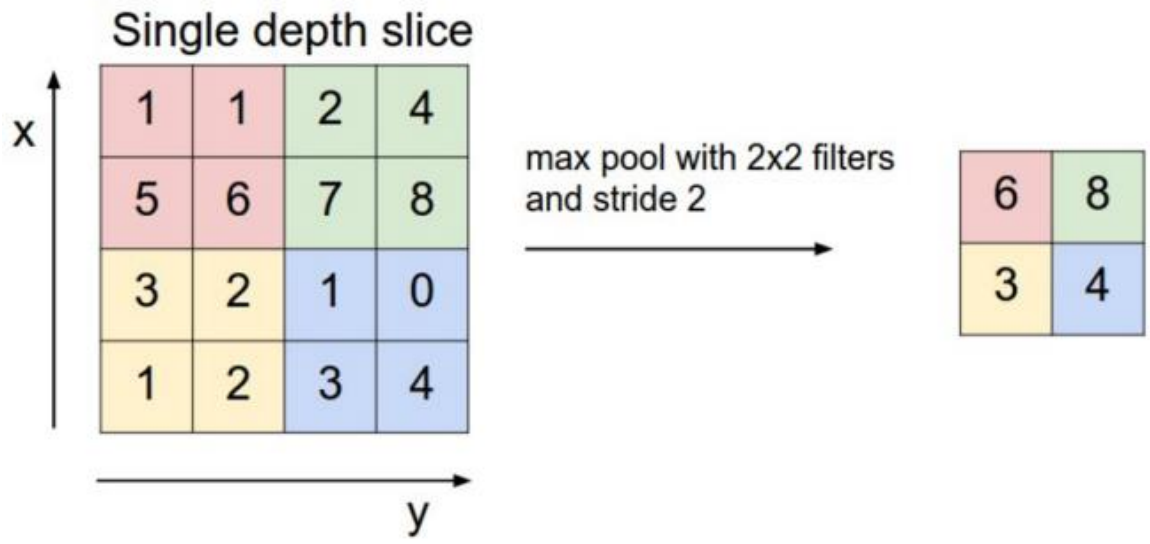

Figure 5. MaxPooling

- **Flattening:** The resulting feature maps are flattened into a one-dimensional vector after the convolution and pooling layers so they can be passed into a completely linked layer for categorization or regression.
- **Fully Connected Layers:** It takes the input from the previous layer and computes the final classification or regression task.

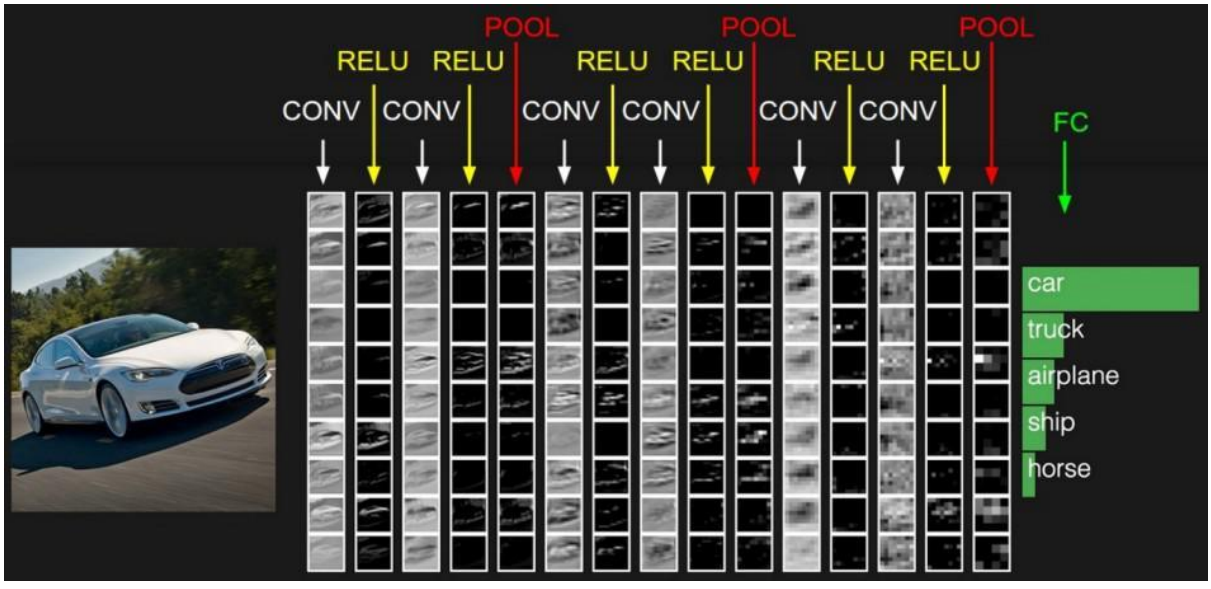

Figure 6. Practical Application of Convolution

- **Output Layer:** The output from the fully connected layers is then fed into a logistic function for classification tasks like sigmoid or softmax which converts the output of each class into the probability score of each class.

CHAPTER-2

LITERATURE SURVEY

2.1 INTRODUCTION

Over the past decade, deep convolutional neural networks (CNNs) have significantly advanced high-level vision tasks like visual recognition, motion analysis, and object segmentation. More recently, CNNs have been applied to low-level vision tasks, such as super-resolution (SR), image denoising, and compression artifact reduction, where they are trained to map low-quality images to high-quality outputs, typically aiming to remove noise or minimize artifacts. While "deeper is better" is a widely accepted principle in high-level vision tasks—evidenced by networks like VGG, GoogleNet, and ResNet achieving substantial breakthroughs—this principle has not shown as much impact in low-level vision tasks. Despite the use of networks with 20 to 30 layers, such as DnCNN [1] and RED-Net, the performance gains in lowlevel tasks have been modest compared to earlier methods. This is because low-level vision tasks rely more on pixel-level features, where depth is less crucial. Instead, statistical priors, like non-local similarity or pixel distribution patterns (e.g., Gaussian noise), play a key role in enhancing the accuracy of these tasks, offering a more effective solution to image degradation issues.

2.2 EXISTING WORK

In CBDNet [2], an asymmetric loss function was employed to enhance the model's ability to generalize to real-world noise scenarios, while also facilitating convenient interactive denoising. For training, they utilized a dataset comprising 400 images from BSD500 [4], 1600 images from Waterloo [11], and 1600 images from the MIT-Adobe FiveK dataset [10]. A batch size of 32 was selected, with each patch being 128 × 128 in size. The model was trained over 40 epochs, starting with a learning rate of 10^{-3} for the first 20 epochs, followed by a learning rate of 5×10^{-4} for fine-tuning during the remaining epochs. The model demonstrated PSNR scores of 30.78 on the SIDD dataset [5] and 38 on the DND dataset. Processing a 512x512 image takes approximately 0.4 seconds.

In their work on RIDNet [3], the authors introduce a CNN-based denoising model specifically designed for both synthetic noise and real-world noisy images. This model is a single-blind denoising network for real noisy images, differing from prior methods. To enhance the network's ability to learn and improve feature extraction, they incorporated a restoration module along with feature attention, which adjusts the channel-wise features by considering the interdependencies between channels. Additionally, they implemented LSC, SSC, and SC mechanisms, allowing low-frequency information to bypass the network, enabling it to focus on learning residuals. The model's architecture consists of four Enhanced Attention Mechanism (EAM) blocks, where most convolutional layers have a kernel size of 3x3, except for the final layer in the enhanced residual block and feature attention units, which utilize a 1x1 kernel. To ensure feature map dimensions remain consistent, the model applies zero-padding to the 3x3 convolutions. Each layer operates with 64 channels, and a downscaling factor of 16 is applied in the feature attention process, reducing the number of feature maps. The model processes a 512x512 image in approximately 0.2 seconds during evaluation and achieved PSNR scores of 31.38, 39.23, and 38.71 on the BSD68 [4], DnD [6], and SIDD [5] datasets, respectively.

The NBNet [7] architecture is built upon a modified UNet framework. NBNet incorporates four encoder and decoder stages, where feature maps are downsampled using strided convolutions in the encoder and upsampled using deconvolutions in the decoder. Skip connections between encoder and decoder stages transfer low-level features, and the primary innovation lies in the introduction of Subspace Attention (SSA) modules within these skip connections. Unlike conventional UNet architectures, where low- and high-level feature maps are directly fused, NBNet leverages SSA modules to project low-level features into a signal subspace guided by upsampled high-level features before fusion. This projection allows the model to better capture global structure information while preserving local details, improving denoising performance. It achieved PSNR scores of 29.16, 39.62 and 39.75 on BSD68 [4], DnD [6] and SIDD [5] datasets respectively.

DANet [2] is the latest architecture introduced for realworld image denoising tasks. Its core concept revolves around unfolding in-camera processing pipelines or learning the noise distribution through a generative adversarial network (GAN). In this dual adversarial framework, three components require optimization: the denoiser (R), the generator (G), and the discriminator (D). To stabilize training, they incorporated the gradient penalty technique from

WGAN-GP [3], ensuring the discriminator adheres to a 1-Lipschitz constraint by adding an additional gradient penalty term, d. DANet includes two hyperparameters—one primarily affecting the performance of the denoiser R, while the other directly impacts the generator G. This architecture achieved PSNR scores of 39.25 and 39.79 on the SIDD and DND benchmark datasets, respectively.

2.3 EVALUATION METRICS

1) **MSE** [8]: The Mean Squared Error(MSE) is a statistical measure that calculates the average of the squared differences between predicted and actual values. It serves as a risk function and is a key metric in empirical risk minimization, particularly in machine learning. The MSE formula takes the mean of the squared errors, where an error is the difference between the observed and predicted values. It can be calculated both within-sample (on the same data used to fit the model) and out of sample (using test data). In matrix form, it can be written as the squared errors summed and averaged over the number of data points.

$$\mathrm{MSE} = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2$$

2) **PSNR** [8]: The Peak Signal-to-Noise Ratio (PSNR) [8] measures the ratio between the maximum possible signal power and the power of the noise that distorts image quality. Typically expressed in decibels (dB), PSNR [8] uses a logarithmic scale due to the wide dynamic range of signals, which reflects the difference between the largest and smallest possible values. It is a common metric for evaluating the quality of image reconstruction in lossy compression codecs, where the signal represents the original data and noise represents distortion or error. PSNR [8] values typically range from 30 to 50 dB for 8- bit data and 60 to 80 dB for 16-bit data in image and video compression. In wireless transmission, a quality loss of 20 to 25 dB is generally acceptable. The PSNR [8] formula is given by:

$$\mathrm{PSNR} = 10 \cdot \log_{10}\left(\frac{\mathrm{MAX}^2}{\mathrm{MSE}}\right)$$

3) **SSIM (Structural Similarity Index Method):** is a perception-based model for evaluating image degradation by analyzing changes in structural information. It measures perceived image and video quality by comparing an original image to its degraded version. SSIM emphasizes the interdependence of spatially close pixels and includes factors like luminance masking (distortions less visible along edges) and contrast masking (distortions less noticeable in textured areas).

Variants of SSIM:

i. Multi-Scale SSIM (MS-SSIM): Evaluates images at different scales, considering luminance, contrast, and structural changes across resolutions for improved subjective performance.

ii. Three-Component SSIM (3-SSIM): Accounts for human visual system's sensitivity to texture over smooth areas by disaggregating images into edges, texture, and smooth regions with weights of 0.5, 0.25, and 0.25 respectively. A 1/0/0 weighting (focusing on edges) aligns closer to human perception.

Formula for SSIM(x, y):

$$\mathrm{SSIM}(x, y) = \frac{(2\mu_x\mu_y + C_1)(2\sigma_{xy} + C_2)}{(\mu_x^2 + \mu_y^2 + C_1)(\sigma_x^2 + \sigma_y^2 + C_2)}$$

SSIM index ranges from -1 to 1, where 1 represents perfect similarity between images.

2.4 DATASETS

1) **SIDD** [5]: In recent years, there has been a significant shift from DSLR and point-and-shoot cameras to smartphone imaging, where the smaller apertures and sensor sizes of smartphones result in higher noise levels. Despite extensive research on smartphone image denoising, the field lacked a comprehensive dataset of real noisy images with corresponding high-quality ground truth. To address this gap, this dataset presents a systematic approach for estimating ground truth for noisy smartphone images, creating a new dataset—the Smartphone Image Denoising Dataset (SIDD)—comprising approximately 30,000 images from 10 scenes under varying lighting conditions, captured using five different smartphone cameras. This dataset

enables benchmarking of denoising algorithms, and it has been demonstrated that CNN-based methods trained on this high-quality dataset outperform models trained with alternative strategies, such as using low-ISO images as proxy ground truth.

2) **BSD** [4]: The Berkeley Segmentation Dataset (BSD) [4] is a widely used collection of images for evaluating segmentation and image processing algorithms, including image denoising. It contains several subsets with varying numbers of images, each providing a rich source of diverse natural images.

i. **BSD68** [4]: This subset consists of 68 images selected from the larger dataset. It is often used as a benchmark for evaluating image segmentation and denoising algorithms due to its manageable size and representative diversity.
ii. **BSD100** [4]: This version includes 100 images, expanding on the BSD68 [4] set. The increased number of images allows for more comprehensive testing and validation of algorithms, offering a broader range of textures, colors, and structures.
iii. **BSD300** [4]: This dataset features 300 images, providing an even more extensive set of natural images. The larger size enhances the dataset's ability to assess the performance of denoising methods across a wider variety of scenarios, making it useful for researchers looking to develop and test robust algorithms. Overall, these datasets serve as standard benchmarks for evaluating the effectiveness of image denoising techniques and help researchers compare their methods against established results in the field.

3) **Urban100** [9]: The dataset consists of 100 high resolution images that capture various urban scenes, including buildings, streets, and other architectural features. These images contain a range of textures, colors, and structures, which make them suitable for testing the robustness of denoising algorithms. The dataset consists of 100 high-resolution images that capture various urban scenes, including buildings, streets, and other architectural features. These images contain a range of textures, colors, and structures, which make them suitable for testing the robustness of denoising algorithms. The dataset includes a wide variety of scenes, ensuring that models trained or tested on Urban100 [9] can generalize well across different types of urban imagery. While the original images are clean, researchers often add synthetic noise (e.g., Gaussian noise) at various levels to create noisy versions for denoising tasks. This setup allows for a controlled evaluation of how well different algorithms can recover the original clean images from their noisy counterparts.

2.5 DRAWBACKS OF EXISTING METHODS

1) **High Computational Complexity and Resource Demand:** State-of-the-art denoising models typically consist of millions of trainable parameters, leading to significant computational demands. While these models perform exceptionally well in controlled laboratory environments, their need for vast datasets and computational resources makes them impractical for many real-world applications.

2) **Inefficiency in Image Processing Speed:** Current state of the art models require 0.2 to 0.4 seconds to denoise a single 512x512 image. This processing time limits their usability in scenarios where real-time or high-speed image processing is essential.

2.6 OBJECTIVES

1) To reduce the computational complexity and resource demand for training and inferring the model.
2) To improve image processing speed
3) To maintain or improve the state-of-the art results while keeping the model efficient.

CHAPTER-3

METHODOLOGY

3.1 INTRODUCTION

In this section, we present the methodology adopted for the blind denoising of images using a Convolutional Neural Network (CNN)-based approach. The proposed model architecture is carefully designed to address the task of image denoising, catering to high-performance requirements in resource-limited environments. Our CNN-based architecture comprises two main parts: a Super Resolution network and a Denoiser network, enabling the model to upscale noisy, low-resolution images before removing noise. By leveraging the structural capabilities of CNN layers, the model efficiently captures essential features for both high-frequency detail recovery and noise suppression, achieving competitive denoising results. The architecture's streamlined design balances computational efficiency with denoising effectiveness, providing a practical solution for image restoration across a variety of noise levels and types without prior information on noise characteristics.

3.2 INPUT LAYER

Input Shape: (112, 112, 3)

The model accepts an input image of size 112 × 112 with three color channels (RGB). This compact image resolution is chosen to reduce computational complexity and memory requirements while ensuring efficiency in model training and inference.

3.3 SUPER RESOLUTION BLOCK

Output Shape: (224, 224, 3)
Trainable Parameters: 110,515

The Super-Resolution Block is responsible for upscaling the input image from (112 × 112) to (224 × 224) by leveraging a series of convolutional layers and residual connections. This

block effectively doubles the spatial resolution of the image and forms the encoder part of the architecture.

- **Initial Convolution Layer:** A Conv2D layer with 32 filters and a kernel size of 3×3 is applied to the input, followed by a LeakyReLU activation. This step extracts low-level features from the image.
- **Residual Blocks:** Two Residual Blocks are employed. Each block consists of two convolutional layers with a kernel size of 3 × 3 and 32 filters, followed by Batch Normalization and LeakyReLU activations. These blocks capture intricate features while addressing the vanishing gradient problem. Skip connections are added to improve feature flow and gradient propagation, enhancing model training.
- **Final Convolution and Upsampling:** The final convolutional layer in the Super-Resolution Block has 64 filters, followed by a Depth-to-Space operation (also known as Pixel Shuffling), which rearranges the tensor to increase its spatial resolution to 224×224. This technique provides an efficient way to upscale the image.
- **Output:** A Conv2D layer with 3 filters reconstructs the RGB image, which now has double the spatial dimensions (224 × 224) compared to the input.

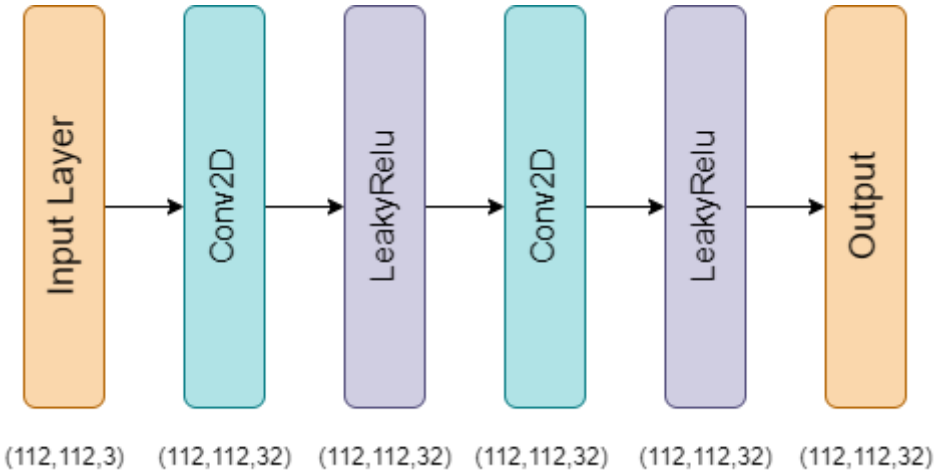

Figure 7. Residual Block

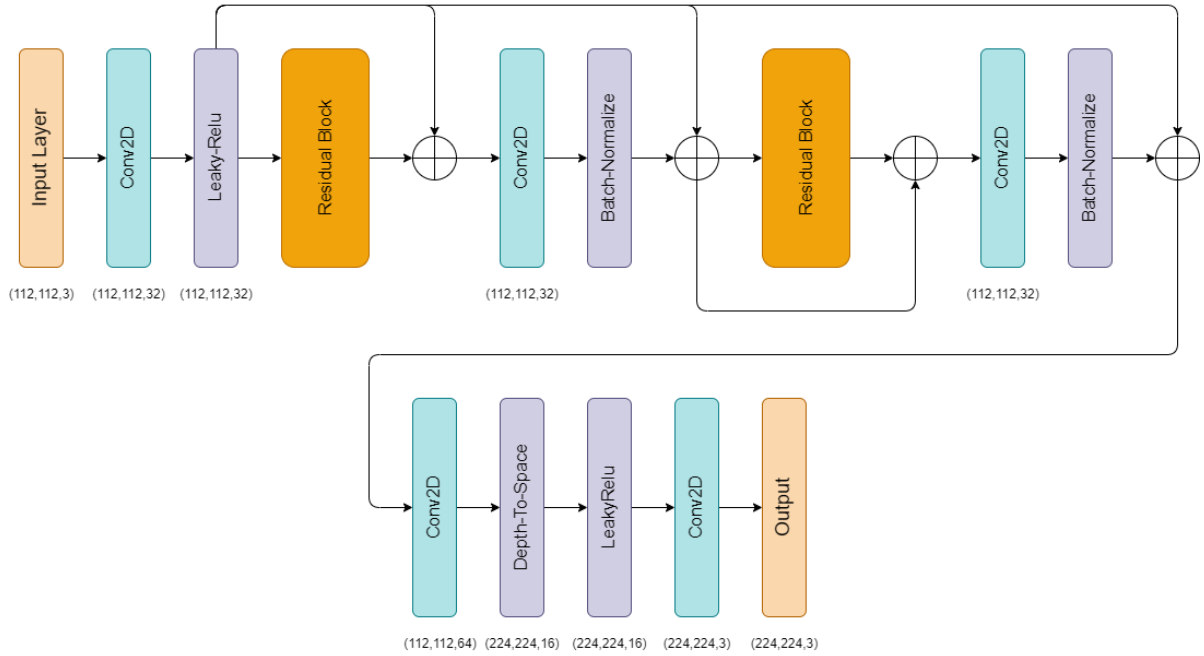

Figure 8. Super Resolution Block

3.4 DENOISER BLOCK

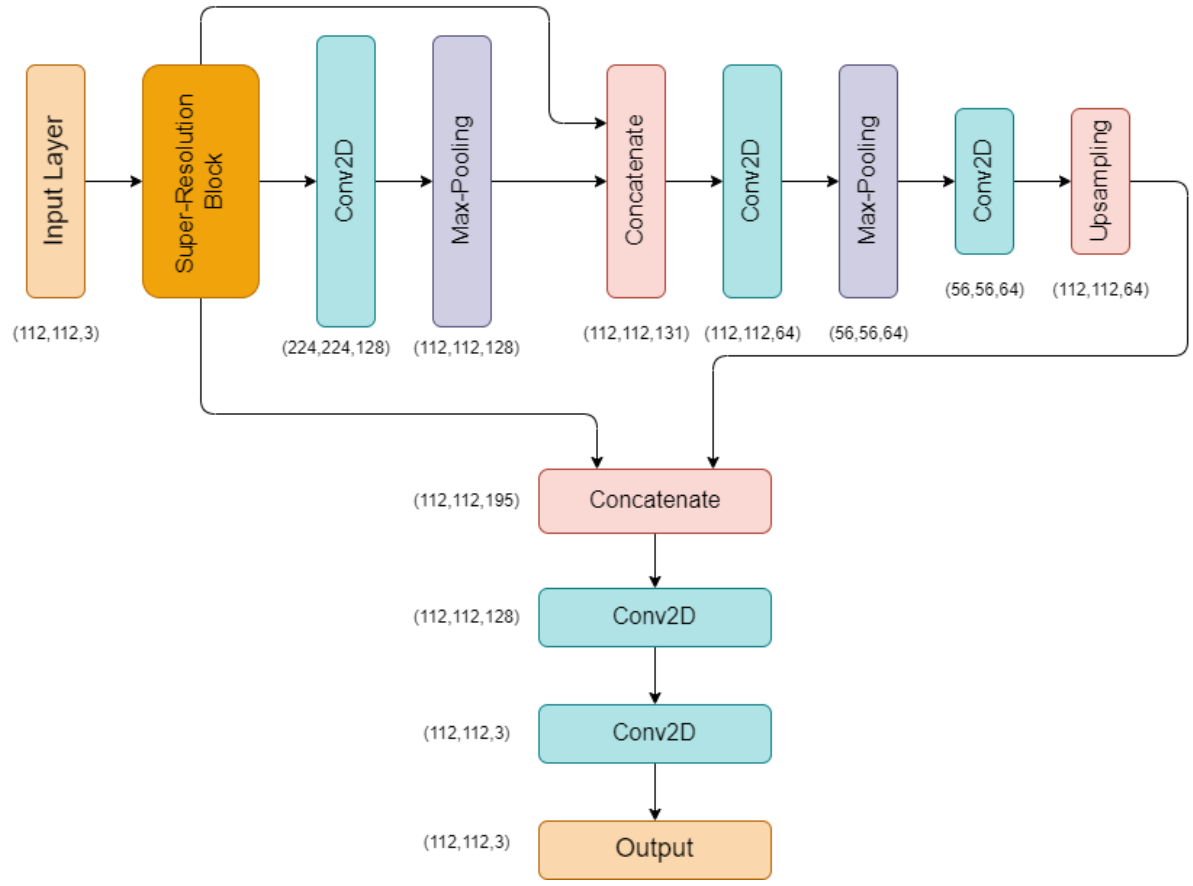

Figure 9. Model Architecture

Output Shape: (112, 112, 3)
Trainable Parameters: 344,259

The Denoiser Block performs the task of noise removal by processing the upscaled image from the Super-Resolution Block. This component can be considered a decoder block, reducing the resolution back to (112 × 112) while preserving important visual features through skip connections and convolution operations.

- Convolution Layers: The first convolution layer in the Denoiser Block has 128 filters, followed by a MaxPooling2D layer that downscales the image to (112 × 112) for processing.
- Concatenation with Input: A skip connection is established where the original input image is concatenated with the down sampled version of the upscaled image, producing an intermediate feature map of size (112×112×131). This fusion ensures the model retains essential features from the initial input.
- Down sampling and Upsampling: – The image is further down sampled using another convolutional layer with 64 filters, followed by another MaxPooling2D operation that reduces the spatial size to (56 × 56). An UpSampling2D layer increases the resolution back to (112 × 112).
- Concatenation with Intermediate Features: Skip connections are introduced again by concatenating the up sampled feature map with previous intermediate layers, producing a feature map of size (112 × 112 × 195).
- Final Convolutions: The concatenated feature map is passed through a convolutional layer with 128 filters and finally reduced to 3 channels (RGB) using a Conv2D layer with 3 filters. The output is the denoised image, reconstructed to its original resolution of (112 × 112).

3.5 WORKING PRINCIPLE

- The architecture leverages the Super-Resolution Block to enhance the resolution of the input image from (112 × 112) to (224×224). This block employs a combination of convolutional layers, residual connections, and depth-to-space transformation to ensure that the upscaled image retains fine details without introducing significant

artifacts.

- Following this, the Denoiser Block takes over, utilizing down sampling and upsampling techniques, combined with skip connections, to effectively remove noise from the upscaled image. The final result is a clean, denoised image at the original resolution of (112 × 112).

- Skip connections play a crucial role in both blocks by preserving critical features from the earlier stages and preventing information loss during down sampling and upsampling.

3.6 SUMMARY

• Total Parameters: 454,902
• Trainable Parameters: 454,774
• Non-trainable Parameters: 128

This architecture is highly optimized for the task of image denoising, leveraging the efficiency of the Super-Resolution Block to work with smaller input patches, significantly reducing the amount of data and time required for training and inference. Furthermore, the Denoiser Block ensures that the model can effectively remove noise while retaining critical image features, providing high-quality denoised images as output.

CHAPTER-4

IMPLEMENTATION

4.1 INTRODUCTION

In this section, we outline the process for implementing the CNN-based architecture used for blind denoising of images. We begin with data preprocessing, where the noisy images are prepared for training by resizing them to a consistent shape and applying normalization to aid in model convergence. Gaussian noise is artificially introduced to simulate realistic image degradation, which helps in enhancing the model's robustness. Next, we detail the model training using TensorFlow and Keras, including architectural choices, hyperparameter tuning, and training strategies. The network is trained with a progressive learning rate schedule and evaluated using PSNR to optimize both denoising accuracy and computational efficiency.

4.2 DATA PREPROCESSING

1) **SIDD** [5]: In our study, we utilized images from the SIDD [5] dataset, resizing them to dimensions of (2576, 1456, 3). From each image, we extracted patches with a resolution of (112, 112, 3). For training, we randomly selected 7,000 patches, while 3,000 patches were allocated for validation. The final 10 images from the dataset were reserved for testing, which, after patching, yielded 2,990 test patches.

2) **Urban100** [9]: In our study, we utilized the Urban100 [9] dataset to extract a total of 80 images for the training set. These images were cropped to a size of 560 × 560 × 3 pixels. Subsequently, we generated patches of size 112×112×3 from each image, resulting in a total of 2,000 patches designated for training with a validation split of 0.1. For the testing phase, we selected 20 remaining images from the dataset, which were similarly cropped to 560×560×3 pixels. Following the patch generation process, we obtained 500 patches of size 112 × 112 × 3 for testing purposes.

4.3 MODEL TRAINING

The Super-Resolution Image Denoiser Network (SRIDNet) was trained using the TensorFlow and Keras frameworks on image patches of size (112, 112, 3). The training process was carried out in three stages, each with progressively reduced learning rates to enhance model performance and stability. Initially, the model was trained for 20 epochs with a learning rate of 0.001.

This was followed by an additional 20 epochs at a reduced learning rate of 0.0001, and a final 10 epochs at a further reduced learning rate of 0.00001. The Adam optimizer was employed to minimize the mean squared error (MSE) loss function, with Peak Signal-to-Noise Ratio (PSNR) used as the evaluation metric. The batch size was set to 16 for all training stages. On the Urban100 dataset, the model required approximately 45 seconds per epoch, while on the SIDD dataset, it required 130 seconds per epoch. These timings reflect the computational complexity and dataset size differences between the two datasets.

CHAPTER-5

RESULTS

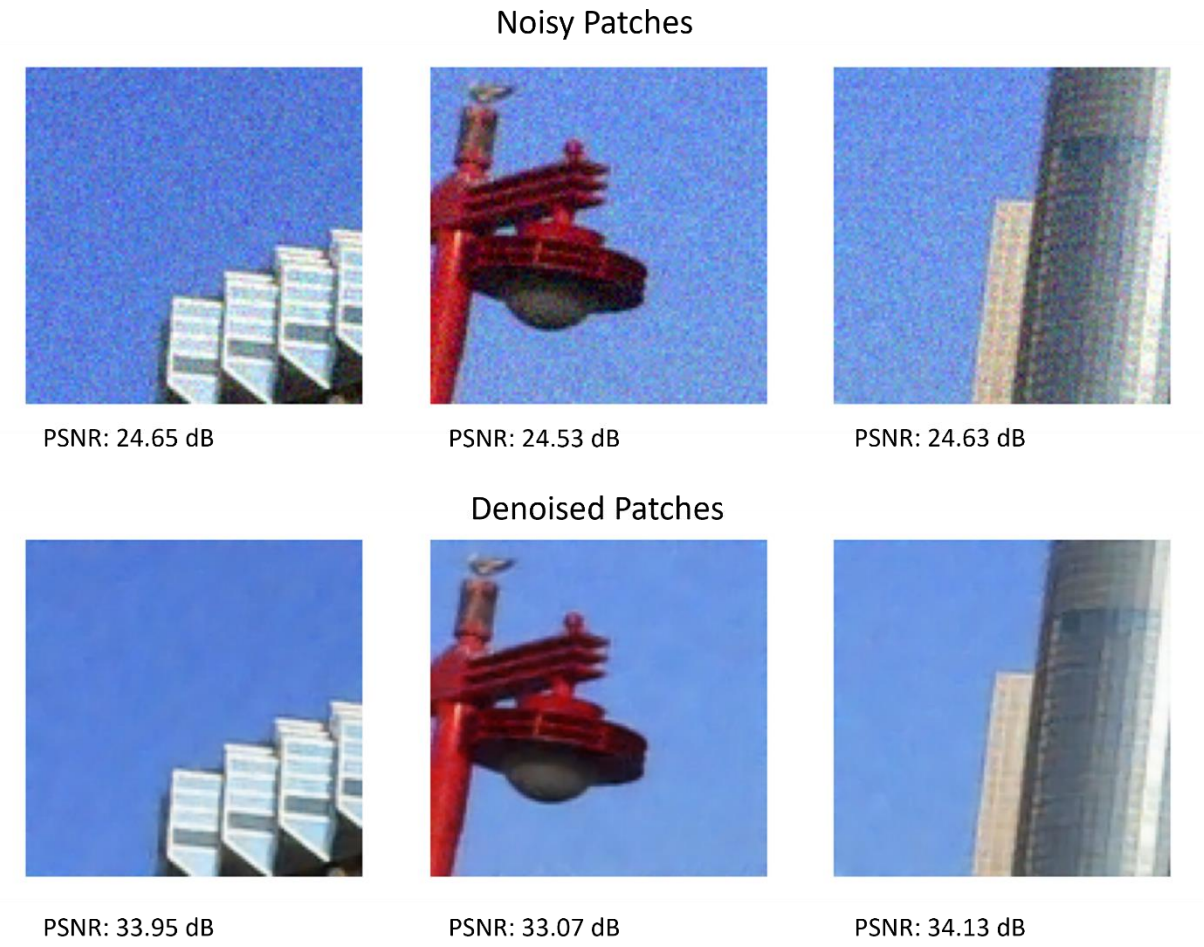

Figure 10. PSNR results on 112 × 112 × 3 image patches (σ = 15) from Urban100 dataset

Figure 11. PSNR results on an entire image (σ = 15) of size 560 × 560 × 3 from Urban100 dataset

The proposed Super-Resolution Image Denoiser Network (SRIDNet) was evaluated on the Urban100 [9] dataset, with the last 10 images used as the test set. The evaluation process was conducted with TensorFlow, and inference was completed in an average time of 20-24 ms per image of size 560×560×3 at noise level (σ = 15). The model's performance was assessed using Peak Signal-to-Noise Ratio (PSNR) as the evaluation metric. The results indicate that SRIDNet achieved an average PSNR of 34.23 dB across the test images, with a maximum PSNR of 35.82 dB.

```
1/1 [==============================] - 0s 24ms/step
1/1 [==============================] - 0s 20ms/step
1/1 [==============================] - 0s 20ms/step
1/1 [==============================] - 0s 23ms/step
1/1 [==============================] - 0s 20ms/step
1/1 [==============================] - 0s 20ms/step
1/1 [==============================] - 0s 20ms/step
1/1 [==============================] - 0s 20ms/step
1/1 [==============================] - 0s 22ms/step
1/1 [==============================] - 0s 19ms/step
Avg PSNR:  tf.Tensor(34.230599650363615, shape=(), dtype=float64) Max PSNR:  tf.Tensor(35.81651203385877, shape=(),
dtype=float64)
```

Figure 12. Avg. PSNR and evaluation time per image in TensorFlow and Keras for 560 × 560 × 3 images at noise level (σ = 15) from Urban100 dataset.

Model	Sigma	PSNR (dB)
DnCNN	15	32.98
	25	30.81
	50	27.59
FFDNet	15	33.83
	25	31.40
	50	28.05
DRUNet	15	34.81
	25	32.60
	50	29.61
SwinIR	15	35.13
	25	32.90
	50	29.82
Ours	15	**34.13**
	25	**32.01**
	50	**28.67**

Table 1. Comparison of Avg. PSNR results with existing models on Urban100 dataset.

The proposed Super-Resolution Image Denoiser Network (SRIDNet) was evaluated on the Urban100 dataset and compared against state-of-the-art denoising models, including DnCNN [1], FFDNet [12], DRUNet [13], and SwinIR [14], across different noise levels (σ = 15, 25, 50). The evaluation metric is Peak Signal-to-Noise Ratio (PSNR) as presented in Table 1.

```
10/10 [==============================] - 1s 104ms/step
10/10 [==============================] - 1s 91ms/step
10/10 [==============================] - 1s 91ms/step
10/10 [==============================] - 1s 91ms/step
10/10 [==============================] - 1s 91ms/step
10/10 [==============================] - 1s 91ms/step
10/10 [==============================] - 1s 91ms/step
10/10 [==============================] - 1s 91ms/step
10/10 [==============================] - 1s 91ms/step
10/10 [==============================] - 1s 91ms/step
tf.Tensor(39.50148010907847, shape=(), dtype=float64)
```

Figure 13. Avg. PSNR and evaluation time of SIDD images in TensorFlow and Keras

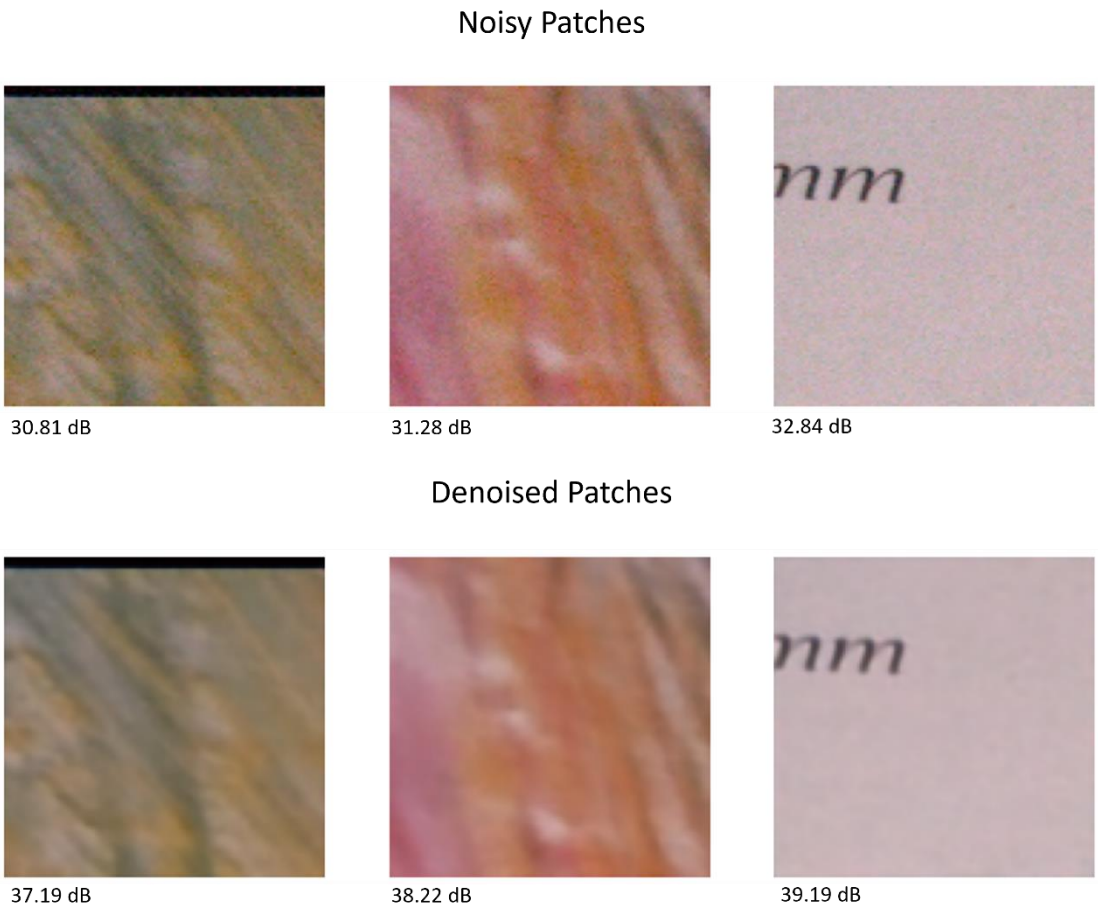

Figure 14. Denoising results on SIDD dataset images

The proposed model was evaluated on the SIDD dataset and benchmarked against several state-of-the-art denoising models, including DANet [16], VDN [15], RIDNet [3], and CBDNet [2].

The model achieved an average PSNR of 39.5 on the SIDD dataset, with an average inference time of 95 ms for images sized 2576 × 1456, which were divided into 299 patches of size 112×112.

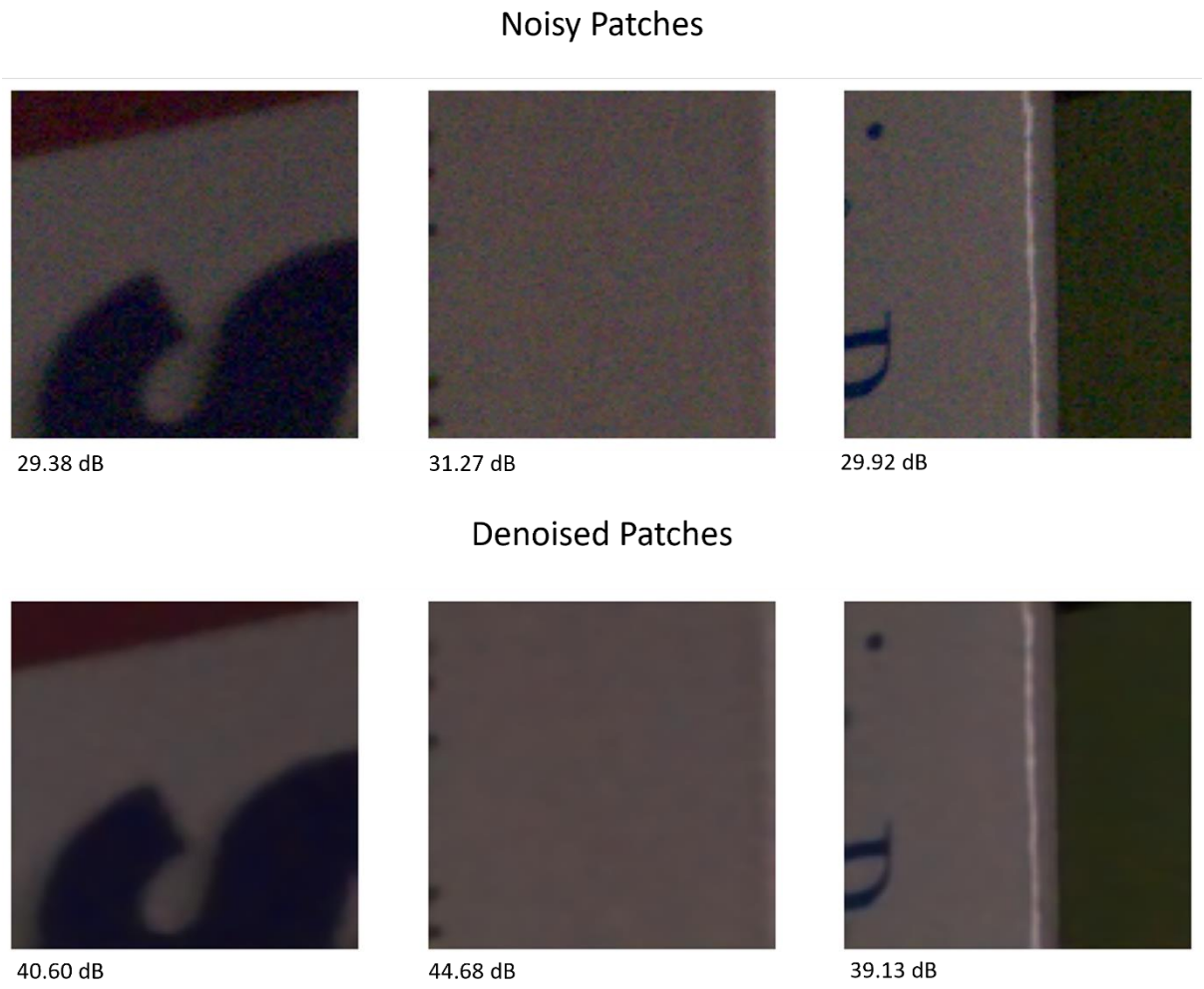

Figure 15. Denoising results on SIDD dataset images

Figures 14. and 15. illustrate the denoising results of lowlight and bright-light images from the SIDD [5] dataset, respectively, using (112 × 112 × 3) image patches. Fig 19. presents a comparison of denoising performance across different models on the SIDD [5] dataset, showing that our model outperformed others by achieving a higher average PSNR. While DnCNN [1] and FFDNet [12]—models of similar size—performed well on images with additive Gaussian white noise (AGWN) from Urban100 [9] dataset, they underperformed on real-world photographic images from the SIDD [5] dataset. In contrast, our model demonstrated robust performance across both datasets with greater efficiency. Table 3. compares the inference times of the models for denoising an image of size 256 x 256 evaluated

in this study. While our model is faster than all models except FFDNet [12], it offers more consistent results on both the Urban100 and SIDD [5] datasets, where FFDNet [12] struggles with real-world images.

Model	**PSNR (dB)**
CBDNet	30.78
RIDNet	38.71
VDN	39.28
DANet	39.47
DnCNN	26.21
FFDNet	29.20
Ours	**39.50**

Table 2. Comparison of Avg. PSNR results with existing models on SIDD dataset.

Model	**Inference Time (s)**
DnCNN	0.0314
FFDNet	0.0071
CBDNet	0.4
RIDNet	0.2
Ours	**0.0208**

Table 3. Comparison of inference time with existing models.

CHAPTER-6

CONCLUSION AND FUTURE SCOPE

In conclusion, this research presents SRIDNet, a novel deep learning architecture for image denoising that effectively addresses the limitations of current state-of-the-art models, such as high computational demands, extensive training data requirements, and slow inference times. By designing a lightweight model that integrates super-resolution and denoising tasks, SRIDNet achieves competitive results with significantly reduced resource consumption. Extensive testing on both synthetic (AGWN) and real-world noisy images from the Urban100 and SIDD datasets demonstrates the model's robustness and efficiency. With PSNR scores of 34.23 on Urban100 and 39.50 on SIDD, SRIDNet delivers near stateof-the-art performance while maintaining the smallest model size and fastest inference time. This balance of efficiency and effectiveness positions SRIDNet as a promising solution for practical image denoising applications, particularly in environments with limited computational resources.

There's a growing need for lightweight models that can denoise images in real-time, especially for applications on mobile devices or embedded systems. Techniques like model quantization, pruning, and knowledge distillation could enhance efficiency without compromising performance. Deploying efficient models in environments with limited computational resources, such as medical devices, is a critical area of focus.

REFERENCES

[1] Kai Zhang, Wangmeng Zuo, Yunjin Chen, Deyu Meng, and Lei Zhang. Beyond a gaussian denoiser: Residual learning of deep cnn for image denoising. IEEE Transactions on Image Processing, 26:3142–3155, 2017.

[2] S. Guo, Z. Yan, K. Zhang, W. Zuo, L. Zhang: Toward Convolutional Blind Denoising of Real Photographs. In: CVPR (April 2019).

[3] S. Anwar, N. Barnes.: Real Image Denoising with Feature Attention. In: ICCV (March 2020).

[4] Stefan Roth and Michael J Black. Fields of experts. IJCV, 2009. 1, 2, 5, 6, 7

[5] Abdelrahman Abdelhamed, Stephen Lin, and Michael S Brown. A highquality denoising dataset for smartphone cameras. In CVPR, 2018. 5, 8

[6] Tobias Plotz and Stefan Roth. Benchmarking denoising algorithms with real photographs. arXiv preprint arXiv:1707.01313, 2017. 5, 7

[7] S. Cheng1, Y. Wang1, H. Huang, D. Liu, H. Fan and S. Liu.: NBNet: Noise Basis Learning for Image Denoising with Subspace Projection. In: CVPR (May 2021).

[8] Sara, U. , Akter, M. and Uddin, M. (2019) Image Quality Assessment through FSIM, SSIM, MSE and PSNR—A Comparative Study. Journal of Computer and Communications, 7, 8-18. doi: 10.4236/jcc.2019.73002.

[9] J.-B. Huang, A. Singh, and N. Ahuja, ”Single image super-resolution from transformed self-exemplars,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR), June 2015.

[10] V. Bychkovsky, S. Paris, E. Chan, and F. Durand, ”Learning photographic global tonal adjustment with a database of input/output image pairs,” in Proc. 24th IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR), 2011.

[11] K. Ma, Z. Duanmu, Q. Wu, Z. Wang, H. Yong, H. Li, and L. Zhang, ”Waterloo Exploration Database: New challenges for image quality assessment models,” IEEE Transactions on Image Processing, vol. 26, no. 2, pp. 1004-1016, Feb. 2017.

[12] Kai Zhang, Wangmeng Zuo, and Lei Zhang. Ffdnet: Toward a fast and flexible solution for cnn-based image denoising. TIP, 2018. 1, 2, 5, 6, 7, 8

[13] Kai Zhang, Yawei Li, Wangmeng Zuo, Lei Zhang, Luc Van Gool, and Radu Timofte. Plug-and-play image restoration with deep denoiser prior. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2021. 1, 2, 7, 8

[14] J. Liang, J. Cao, G. Sun, K. Zhang, L. Van Gool, and R. Timofte, ”SwinIR: Image Restoration Using Swin Transformer,” arXiv preprint arXiv:2108.10257, 2021.

[15] Z. Yue, H. Yong, Q. Zhao, D. Meng, and L. Zhang, ”Variational denoising network: Toward blind noise modeling and removal,” in Advances in Neural Information Processing Systems 32, H. Wallach, H. Larochelle, A. Beygelzimer, F. d’Alche-Buc, E. ´ Fox, and R. Garnett, Eds. Curran Associates, Inc., 2019, pp. 1690- 1701. Available: http://papers.nips.cc/paper/8446-variational-denoisingnetwork-toward-blind-noise-modeling-and-removal.pdf.

[16] Z. Yue, Q. Zhao, L. Zhang, and D. Meng, ”Dual adversarial network: Toward real-world noise removal and noise generation,” in Proc. Eur. Conf. Comput. Vis. (ECCV), Aug. 2020.