The document has 579 words, which can be summarized and formatted to fit within 4 pages. Here is a condensed version:

---

# Denoising Project Explanation

## Introduction

This project involves a denoising task using convolutional neural networks (CNNs) to remove noise from images. The primary steps include setting up directories, loading images, normalizing them, creating datasets, defining a CNN model, and training the model.

## Directory and Data Setup

1. \*\*Mounting Google Drive\*\*: Use `drive.mount('/content/drive')` to access data stored in Google Drive.

2. \*\*Navigating to Notebook Directory\*\*: Change the working directory to where the notebook is stored.

3. \*\*Test Data Directory\*\*: Set up paths to the high-quality (HQ) and low-quality (LQ) image directories.

4. \*\*Finding Image Paths\*\*: Use `glob.glob` to find all PNG images in the specified directories.

## Loading and Normalizing Images

1. \*\*Image Pair Verification\*\*: Ensure equal numbers of HQ and LQ images using an assert statement.

2. \*\*Loading Image Pairs\*\*: Use `cv2.imread` to read images and store them as NumPy arrays.

3. \*\*Normalization\*\*: Convert image pixel values to the range [0.0, 1.0] for better training performance.

## Dataset Preparation

1. \*\*Splitting Dataset\*\*: Use `train\_test\_split` to divide images into training, validation, and test sets.

2. \*\*Convert to NumPy Arrays\*\*: Convert lists of images into NumPy arrays for efficient computation.

3. \*\*ImagePairDataset Class\*\*: Define a PyTorch dataset class to handle image pairs.

## Data Loaders

1. \*\*Create Data Loaders\*\*: Use `DataLoader` to create data loaders for training, validation, and testing, specifying batch sizes and shuffling options.

## Model Definition

1. \*\*DenoisingCNN Class\*\*: Define a CNN with three convolutional layers, each followed by ReLU activation functions. The final layer reconstructs the denoised image.

2. \*\*Initialization\*\*: Initialize model components including the network, loss function (`nn.MSELoss`), and optimizer (`optim.Adam`).

## Training the Model

1. \*\*Training Loop\*\*: Train the model over several epochs, computing the loss and updating model parameters using backpropagation.

2. \*\*Validation\*\*: Evaluate model performance on the validation set to monitor overfitting and adjust hyperparameters if necessary.

## Evaluation

1. \*\*Test Performance\*\*: After training, evaluate the model on the test set to determine its generalization capability.

2. \*\*PSNR Score\*\*: Calculate the Peak Signal-to-Noise Ratio (PSNR) to quantify the quality of the denoised images. The final PSNR score achieved is 26.158963424072354.

---

This summary provides a concise overview of the denoising project, highlighting key steps and results, including the PSNR score. The detailed processes of data handling, model definition, and training are essential for understanding the project's workflow and outcomes.