**VLG Summer Open Project 2024**

**Denoising Image**

**By- Nimish Shukla**

### Introduction

The objective of this project is to develop an image denoising algorithm using deep learning techniques, with a particular focus on convolutional neural networks (CNNs). Image denoising is a critical preprocessing task in various image processing applications, aiming to remove noise from images while preserving essential details.

### Dataset Information

Our dataset comprises pairs of images, each containing a noisy image and its corresponding clean version. Ensuring that images in both folders are of the same size and properly aligned is crucial for the model to effectively learn the mapping from noisy to clean images.

* **Noisy Images Folder**: 'C:/Users/Nimish/Downloads/low'
* **Clean Images Folder**: 'C:/Users/Nimish/Downloads/high'

### Modules and Libraries Used

* **os**: Handles directory and file operations.
* **cv2**: Provides image processing functions.
* **NumPy**: Supports array and matrix operations.
* **sklearn.model\_selection**: Splits data into training and test sets.
* **TensorFlow**: Facilitates deep learning model creation and training.
* **tensorflow.keras.layers, models**: Defines neural network layers and models.
* **torch.utils.data**: Manages dataset and data loading in PyTorch.
* **torchvision.transforms**: Applies data transformations for image processing.
* **PIL.Image**: Handles image opening and manipulation.
* **skimage.metrics**: Calculates image quality metrics like PSNR.

### Specification

* **Framework**: TensorFlow/keras
* **Layers**: Multiple convolutional layers with activation functions, and upsampling layers to restore image dimensions to their original size.
* **Optimizer**: Adam optimizer
* **Loss function**: Binary-cross entropy

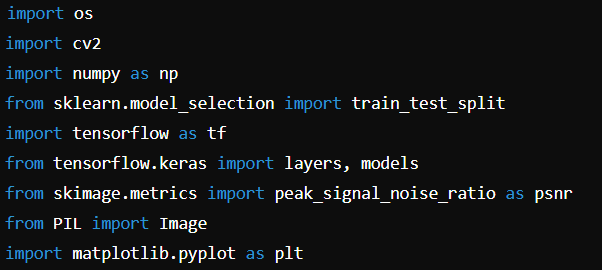
### Model Architecture

This project implements a convolutional neural network (CNN) architecture, well-suited for image processing tasks due to its ability to capture spatial hierarchies within images. Our CNN model includes:

* **Convolutional Layers (Conv2D layers with ReLU activation)**: Extract features from the input images.
* **Activation Functions**: Use Leaky ReLU to avoid the "dying ReLU" issue and enhance gradient flow.
* **Pooling Layers (MaxPooling2D layers for downsampling)**: Downsample feature maps to reduce spatial dimensions and computational load.
* **Upsampling Layers**: Increase the spatial dimensions of the feature maps to match the original image size.

### Code Snippets for Model Creation

#### 1. Load the Necessary Libraries



#### 2. Load Images from the Folder

The function load\_images\_from\_folder reads images from a specified folder, processes them, and returns a list of resized, normalized images.

A computer screen shot of a program

Description automatically generated

#### 3. Prepare Datasets

This code sets up the data pipeline for training and testing an image denoising model using TensorFlow.

A screenshot of a computer program

Description automatically generated

#### 4. Create and Compile the Model

This code defines and compiles a convolutional neural network (CNN) model for image denoising.



#### 5. Training the Model

This line of code trains the denoising model using the training dataset and evaluates its performance on the validation dataset.



#### 6. Results

Evaluate the training progress and performance.

A computer screen shot of a program code

Description automatically generated

### Summary of Results

The training log shows the progression of the denoising model over 50 epochs, starting with a high error that decreases over time. By the 50th epoch, both training and validation losses significantly decrease, suggesting effective learning and generalization. The close values of training and validation losses indicate minimal overfitting. The average PSNR value of approximately 17.3 reflects the model's denoising capability.