VLG SUMMER PROJECT

IMAGE DENOISING REPORT

Introduction:

About the architecture used:

Architecture used is zero_dce (Zero-reference deep curve estimation). deep learning model designed for low-light image enhancement. It was introduced to address the challenges of enhancing images taken in low-light conditions without the need for paired or reference images during training

Specifications:

Input: (200, 300, 3) - representing images of size 200x300 pixels with 3 color channels i.e. RGB.

Layers:

- The model consists of seven convolutional layers.
- Each layer has:
  - Conv2d: A 2D convolution layer.
  - ReLU: A ReLU activation function.
- The final layer includes a tanh activation to ensure the output is within the desired range.

Output:

- The output is an enhanced image, where each pixel's intensity is adjusted based on the learned enhancement curves.

Achieved PSNR:

PSNR achieved was 15.1032

DETAILS ABOUT THE PROJECT:

```python
class AugmentedDataset(Dataset):
    def __init__(self, noisy_images, clean_images, transform=None):
        self.noisy_images = noisy_images
        self.clean_images = clean_images
        self.transform = transform

    def __len__(self):
        return len(self.noisy_images)

    def __getitem__(self, idx):
        noisy_img = self.noisy_images[idx]
        clean_img = self.clean_images[idx]

        if self.transform:
            seed = np.random.randint(2147483647)
            torch.manual_seed(seed)
            noisy_img = self.transform(noisy_img)
            torch.manual_seed(seed)
            clean_img = self.transform(clean_img)

        return noisy_img, clean_img
```

Data augmentation has been done for enhancing the model.

```python
class ZeroDCE(nn.Module):
    def __init__(self):
        super(ZeroDCE, self).__init__()
        self.relu = nn.ReLU(inplace=True)

        self.e_conv1 = nn.Conv2d(3, 16, 3, 1, 1, bias=True)
        self.e_conv2 = nn.Conv2d(16, 16, 3, 1, 1, bias=True)
        self.e_conv3 = nn.Conv2d(16, 16, 3, 1, 1, bias=True)
        self.e_conv4 = nn.Conv2d(16, 16, 3, 1, 1, bias=True)
        self.e_conv5 = nn.Conv2d(16, 16, 3, 1, 1, bias=True)
        self.e_conv6 = nn.Conv2d(16, 16, 3, 1, 1, bias=True)
        self.e_conv7 = nn.Conv2d(16, 3, 3, 1, 1, bias=True)

    def forward(self, x):
        x1 = self.relu(self.e_conv1(x))
        x2 = self.relu(self.e_conv2(x1))
        x3 = self.relu(self.e_conv3(x2))
        x4 = self.relu(self.e_conv4(x3))
        x5 = self.relu(self.e_conv5(x4))
        x6 = self.relu(self.e_conv6(x5))
        x_r = torch.tanh(self.e_conv7(x6))

        x = x + x_r
        return x
```

Defining the zero_dce model with 7 convolutional layers. Each layer increases the channels.

Activation function used is ReLU.

The network aims to enhance the input image x by predicting an improvement (x_r) that is added back to the original, resulting in enhanced quality

```python
for epoch in range(40):
    model.train()
    epoch_loss = 0
    for batch in dataloader:
        noisy, clean = batch
        noisy, clean = noisy.to(device), clean.to(device)
        optimizer.zero_grad()
        denoised = model(noisy)
        loss = mixed_loss(denoised, clean)
        loss.backward()
        optimizer.step()
        epoch_loss += loss.item()
    scheduler.step()
    print(f'Epoch {epoch + 1}, Loss: {epoch_loss / len(dataloader):.4f}')

# Evaluate the model
model.eval()
psnr_values = []
with torch.no_grad():
    for batch in dataloader:
        noisy, clean = batch
        noisy, clean = noisy.to(device), clean.to(device)
        denoised = model(noisy)
        psnr = 10 * torch.log10(1 / mse_criterion(denoised, clean))
        psnr_values.append(psnr.item())
print(f'Average PSNR: {np.mean(psnr_values):.4f}')
```

Model has been trained and PSNR is evaluated

ARCHITECTURE: