SUMMER OPEN PROJECT

IMAGE DENOISING

This project focuses on image denoising using Deep Learning.

The data set consist of noisy and clean images.

The average PNSR of the noisy images is around 7.7.

```
# Calculate PSNR for x
psnr_values = [psnr(y[i], x[i]) for i in range(len(y))]
average_psnr = np.mean(psnr_values)
print(f'Average PSNR: {average_psnr}')
```

```
Average PSNR: 7.765406973920432
```

The model is inspired by RIDnet.

The architecture of the model contains many layers –

1) Channel Attention Layer

   This layer is designed so that important features are recognised and less useful ones are neglected or supressed.

   It consists of ‘fc1’ and ‘fc2’ which are two convolution layers. fc1 has ReLU activation function. This layer functions to reduce the number of channels of image.

   fc2 layer restores number of channels to original.

   After this forward pass with call method is used.

   In this global max pooling and global and fc1 and fc2 are used in both pooling and later addition of both of these layers is used with sigmoid as activation function.

```python
class ChannelAttention(Layer):
    def __init__(self, filters, ratio=16):
        super(ChannelAttention, self).__init__()
        self.filters = filters
        self.ratio = ratio
        self.avg_pool = layers.GlobalAveragePooling2D()
        self.max_pool = layers.GlobalMaxPooling2D()
        self.fc1 = layers.Conv2D(filters // ratio, 1, activation='relu', padding='same')
        self.fc2 = layers.Conv2D(filters, 1, padding='same')

    def call(self, x):
        avg_out = self.fc2(self.fc1(tf.expand_dims(tf.expand_dims(self.avg_pool(x), 1), 1)))
        max_out = self.fc2(self.fc1(tf.expand_dims(tf.expand_dims(self.max_pool(x), 1), 1)))
        return x * tf.sigmoid(avg_out + max_out)
```

2) Spatial Attention Layer

This layer is built to focus on important location of the image. It boosts performance of the Neural network This layer concatenates the average pooled and max polled feature maps along the channel axis to create a new feature map. At last attention is applied with a input tensor x which is multiplied to feature attention map obtained. This is done to focus more on important features and regions in images. This done again by call method.

```python
class SpatialAttention(Layer):
    def __init__(self, kernel_size=7):
        super(SpatialAttention, self).__init__()
        self.conv = layers.Conv2D(1, kernel_size, activation='sigmoid', padding='same')

    def call(self, x):
        avg_out = tf.reduce_mean(x, axis=-1, keepdims=True)
        max_out = tf.reduce_max(x, axis=-1, keepdims=True)
        return x * self.conv(tf.concat([avg_out, max_out], axis=-1))
```

After these layers a conv block function is defines for input and optimal

At last Residual learning is used. It helps in learning the residual features compared to clean image of that of noisy image.

```
def residual_block(x, filters):
    shortcut = x
    x = conv_block(x, filters, 3, 1, activation='relu')
    x = conv_block(x, filters, 3, 1, activation=None)
    x = ChannelAttention(filters)(x)
    x = SpatialAttention()(x)
    x = layers.Add()([x, shortcut])
    x = layers.Activation('relu')(x)
    return x
```

All this is used for building model. The input variable in the is shape of image and the number of residual blocks which are used in residual learning (which I have taken as 10).

I have used the method of stacked generalization to strain the model. I have used the same model two times but data input in both the sub-models are different to increase efficiency.

Input of the model is in format of RGB 256,256.

The input of the first model is 256,256,3 image which is broken down into 16 patches. This is achieved using library known as patchify. Both clean and noisy dataset are patched.

The output obtained from the first model is broken down into 64 patches using patchify. Now since clean image imported first has 16 patches, we have to make a new variable with 64 patches. This data is now used to train the model again.

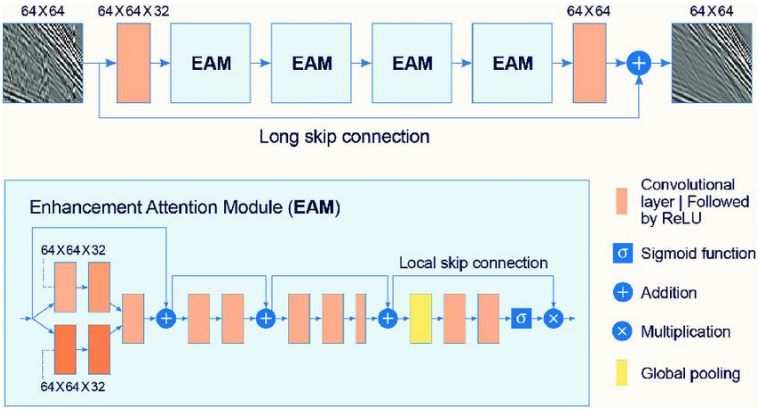
While training I have also used Early stopping and Reduced learning rate to optimize the models. Batch size is taken 16 and a validation split of 10% is taken. Each model runs 30 epochs. Adam optimizer is used.

Final average PSNR obtained is 23.45 approximately.

```
# Calculating psnr for final denoised image i.e. x_denoised_2
psnr_values = [psnr(y[i], x_denoised_2[i]) for i in range(len(y))]
average_psnr = np.mean(psnr_values)
print(f'Average PSNR: {average_psnr}')
```

```
Average PSNR: 23.45090133887874
```

Below is the architecture of RIDnet.



Here EAM is same as Spatial Attention layer mentioned above.

Also I tried to train model on auto-encoders but since PSNR was so didn’t gone for that model but still its resource is mentioned below. The average PSNR obtained from this method was 20.

Potential Improvements

The field of image denoising is constantly evolving. One recent advancement is the use of GANs, which has shown

remarkable results in denoising. More advanced models like HINet, Uformer32, and MIRNet have demonstrated superior performance in image denoising compared to models solely designed for this purpose.

Reference Links

https://www.tensorflow.org/api_docs/python/tf/all_symbols

https://arxiv.org/pdf/1904.07396v2.pdf

https://keras.io/examples/vision/autoencoder/

https://pypi.org/project/patchify/

https://arxiv.org/abs/1807.06521v2

https://arxiv.org/pdf/2009.07485

Project By -- Prahas Tambe

22117104 , ME