EMBEDDED SYSTEMS (EEL3090)
PROJECT REPORT

ES16 - LEARNED IMAGE COMPRESSION

By
Adrika Kunjumon (B21EE003)
Varsha Balamurali (B21EE074)

- **Title:** Learned image compression on a small embedded system

- **Basic theory:** A Convolutional Neural Network (CNN) autoencoder is a type of neural network used for unsupervised learning tasks, such as image compression. It consists of two main parts: the encoder and the decoder.

  The CNN autoencoder encodes the input image into latent space. CNN encoders have convolutional and pooling layers. These layers gradually reduce the input image size while adding channels or features.

  The compressed representation obtained from the encoder is often referred to as the latent space or bottleneck layer. This layer contains the essential features of the input image in a reduced format.

  The CNN autoencoder decoder reconstructs the input image from the encoder's compressed representation. It has convolutional layers followed by upsampling or deconvolutional layers. The layers gradually increase the compressed representation's spatial dimensions while decreasing the number of channels, producing an output image that matches the input.

- **Environment/Libraries used:** Python, tensorflow keras, matplotlib, numpy

- **Dataset specifications:** Dataset used is ‘Kodak dataset’. It consists of 24 high-resolution images. The RGB images are of sizes 512x762 as well as 762x512.

- **Our custom model:** Preprocessing consisted of resizing the images to a standard size like 512x768x3, by flipping certain images to ensure that all images fed into the neural network have the same dimensions.

  We also normalized the pixel values to the range [0,1] by dividing by 255 to help stabilize and speed up the training of the CNN and ensure the model is less sensitive to variations in the input data scale.

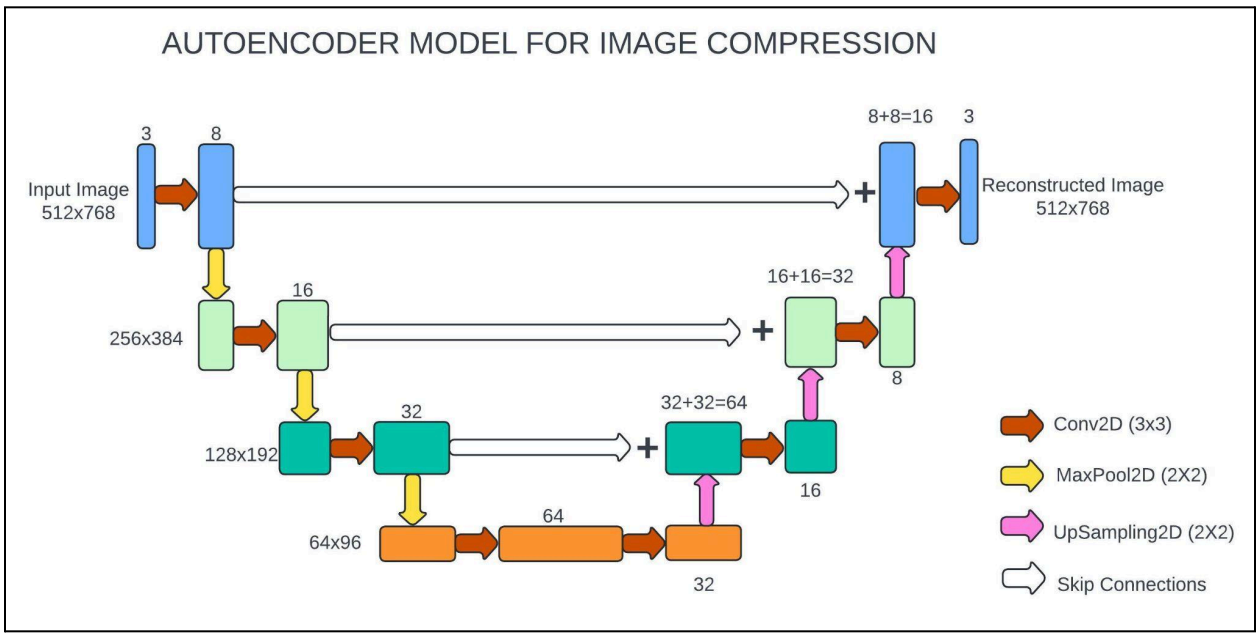
  In our custom CNN model, we have used 3 commonly used types of layers. Firstly, we use the convolutional layers fundamental for feature extraction in CNN to help capture spatial patterns in the input images. Here, we have used Conv2D layers with a filter size of 3x3. Secondly, we use MaxPooling layers for down-sampling and spatial dimensionality reduction. They help in retaining only the most important information of the 2D image. Here, we have used MaxPooling2D layers with stride 2 such that reduces the spatial dimension of that specific layer totally by 4, by reducing each the width and height of the layer by 2. Thirdly, UpSampling layers are used for upscaling the spatial dimensions of the feature maps and to recover the spatial information lost during down-sampling. These 3 types of layers are essential in constructing an autoencoder CNN model.

  The main highlight of our custom model is the addition of skip connections using the Concatenate function. Our model architecture is inspired by the U-Net autoencoder model, which is well-known for its effectiveness in image segmentation tasks. The skip

connections help preserve spatial information and gradients during training, enabling the model to reconstruct high-quality images with fine details.

The importance of skip connections lies in their ability to facilitate the flow of information between different parts of the network, allowing the decoder to leverage features learned by the encoder at multiple scales. This helps mitigate the vanishing gradient problem and enables the model to better capture and reconstruct fine details in the output images. Overall, skip connections contribute to improved performance and faster convergence during training.

Below we have the diagram representation of our custom autoencoder model that shows the dimensions of the feature maps at each stage and specifically shows the skip connections from corresponding layers from the encoder submodel to the decoder submodel.



Model Size Achieved: 214.73 KB

- **Training:** We used the Adam optimizer for training due to its adaptive learning rate capabilities and momentum. It helps converge faster and handle sparse gradients effectively, making it suitable for training deep neural networks like autoencoders. We experimented with different loss functions such as MSE (Mean Squared Error) and BCE (Binary Crossentropy) but got the best results with the MAE (Mean Absolute Error). This must be because it measures the average absolute difference between corresponding pixels of the original and reconstructed images, providing a more robust measure of dissimilarity.

- **Comparison:**

  ➔ JPEG PSNR: 32.1738
  ➔ Model PSNR: 6.8446

  ➔ JPEG SSIM: 0.8996
  ➔ Model SSIM: 0.0157

  ➔ JPEG Compression Ratio: 0.0682
  ➔ JPEG2000 Compression Ratio: 0.7382
  ➔ Model Compression Ratio: 0.3333

- **Conclusion:**
  First, due to computational resource limitations, we were constrained in the complexity and size of the model architecture. As a result, our model lacks some of the advanced features and optimizations found in state-of-the-art compression algorithms like JPEG. Additionally, the availability and diversity of training data posed challenges during model training. Its limited size and variability impacted the generalization ability of our model. In conclusion, our compact deep learning-based image compression model, sized at just 215 KB, demonstrates commendable compression performance metrics as discussed above.