Multiscale Image Compression with Attention-Enhanced Discrete Cosine Transform Models

B.Tech Project Report

Submitted by

Kovidh Pothireddy (B21AI026) , Oppangi Poojita(B21CS055)

Under the Supervision

Of

Prof . Binod Kumar

Department of Computer Science Engineering

Indian Institute of Technology Jodhpur

November , 2024

Abstract

Recently, deep learning-based image compression has made significant progress, achieving better rate-distortion (R-D) performance than traditional methods like JPEG and JPEG2000. However, maintaining a balance between performance and computational complexity remains a challenge. In this paper, we propose an efficient and effective image compression framework that achieves superior R-D performance with lower complexity. First, we introduce a multi-frequency channel attention (MFCA) module that enhances the model's ability to capture and reduce spatial correlations in latent representations. Second, we develop a perceptual quality enhancement module (PQF) to reduce quantization errors and improve the visual quality of reconstructed images. Third, we employ a Pyramid Multi-Convolution Network (PMCN) module to capture multi-scale features and further reduce spatial correlation. Additionally, we design an importance map network to adaptively allocate bits to different regions of the image, optimizing the bit rate. The encoder uses multiple stages of PMCN blocks to capture complex features, while the decoder uses a single stage to reduce decoding complexity. Experimental results on the Kodak dataset show that our method achieves higher PSNR and SSIM values with competitive bit rates compared to JPEG and JPEG2000. The encoding and decoding times are significantly reduced, making our method suitable for real-time applications.

I. Introduction

Deep learning has achieved remarkable success in image compression by leveraging neural network architectures for transform, quantization, and entropy coding within an autoencoder framework. Various methods, such as GDN, residual blocks, and attention modules, have been introduced to find compact and efficient image representations. However, challenges persist in balancing rate-distortion (R-D) performance with computational complexity. Although models like GLLMM outperform traditional methods like H.266/VVC in quality, they often suffer from high encoding and decoding times.

In response to these limitations, we propose an efficient image compression framework that combines Multi-Frequency Channel Attention (MFCA) and Pyramid Multi-Convolution Network (PMCN) modules. The MFCA module captures multiple frequency components using the Discrete Cosine Transform (DCT), enhancing compression efficiency by reducing spatial correlations. Meanwhile, the PMCN module uses multi-scale convolutional layers to capture detailed and global information, which is crucial for high-quality reconstruction. We also introduce an adaptive importance map to allocate bits to important image regions selectively.

Further, a 2D post-quantization filter (PQF) reduces quantization errors by applying a pre-trained filter to the latent representation. Experimental results on the Kodak dataset demonstrate that our approach achieves state-of-the-art PSNR and SSIM scores with competitive bit rates, reduced complexity, and real-time suitability, establishing it as an effective alternative to current image coding methods.

II. Methodology

The proposed CompressionAutoencoder model employs a combination of techniques to achieve efficient image compression. The Multi-Frequency Channel Attention (MFCA) module uses the Discrete Cosine Transform (DCT) to decompose the image into different frequency components, and then applies a channel attention mechanism to emphasize important features. This is followed by the Pyramidal Multi-Scale Convolutional Networks (PMCN) module, which processes the input image at different scales using multiple convolutional layers with different kernel sizes.

The Core Encoder and Decoder are the primary components of the CompressionAutoencoder model. The Core Encoder is responsible for encoding the input image into a compact feature representation, while the Core Decoder reconstructs the image from the compressed feature representation. The importance map, generated using a convolutional layer and sigmoid activation function, is used to weight the encoded features before quantization, ensuring that more important features are preserved during compression.

The quantization and entropy coding components are crucial for reducing the bit rate of the compressed image. The Quantizer module adds uniform noise to the features during training to improve gradient approximation, and performs hard quantization during inference. The Arithmetic Encoder and Decoder simulate the entropy coding process, converting the importance map into a compressed representation and reconstructing it from the compressed representation. The Post-Quantization Filtering (PQF) module enhances the quality of the quantized features using a residual block with two convolutional layers and ReLU activation functions.

The CompressionAutoencoder model is trained to minimize a combined loss function that includes reconstruction loss, perceptual loss, rate loss, and quantization loss. The model is trained using an optimization algorithm, such as Adam, to minimize the loss function and balance compression efficiency and perceptual quality. The performance of the model is evaluated using various metrics, including Peak Signal-to-Noise Ratio (PSNR), Multiscale Structural Similarity (MS-SSIM), and Bit Rate.

III. Implementation

Network Architecture Overview

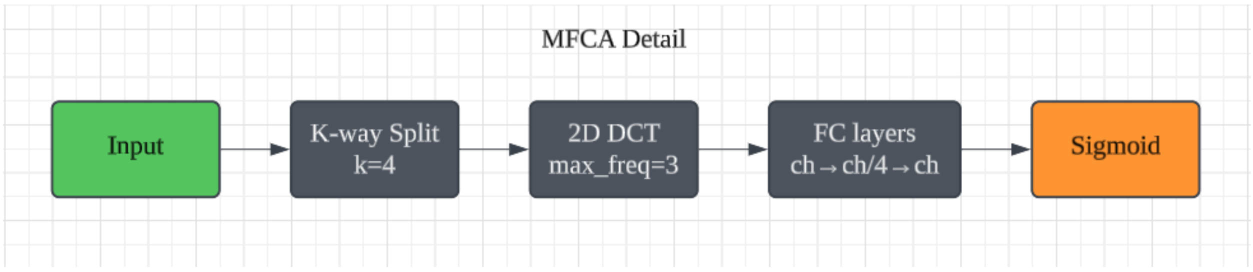
The given deep learning architecture integrates several advanced techniques designed for image restoration, enhancement, and compression. By combining multi-frequency attention, multi-scale convolutional learning, and image compression methods, the model aims to generate high-quality images while handling compression tasks efficiently. Below is an in-depth explanation of each major component:

1. Multi-Frequency Channel Attention (MFCA)

Purpose: MFCA enhances the model's ability to focus on the most informative frequency components of an image, improving its performance on tasks like image restoration, denoising, and enhancement.

Core Components:

- **Discrete Cosine Transform (DCT)**: DCT is used to convert the input feature map from the spatial domain into the frequency domain. This transformation helps to separate the image into various frequency components, enabling the model to focus on the most relevant ones for the task.
    - *Why DCT?* By isolating low, medium, and high-frequency components, the model can better understand patterns at various scales, allowing it to enhance the image's visual quality.
- **Attention Mechanism**: After the frequency components are extracted via DCT, the model applies a fully connected layer (FC) to generate attention scores. These scores are then used to scale the original feature map, ensuring the network prioritizes the most important frequencies.
    - *How attention works*: The attention mechanism dynamically adjusts the emphasis on different frequency components during training, effectively allowing the model to learn which frequency patterns are most valuable for a given task.



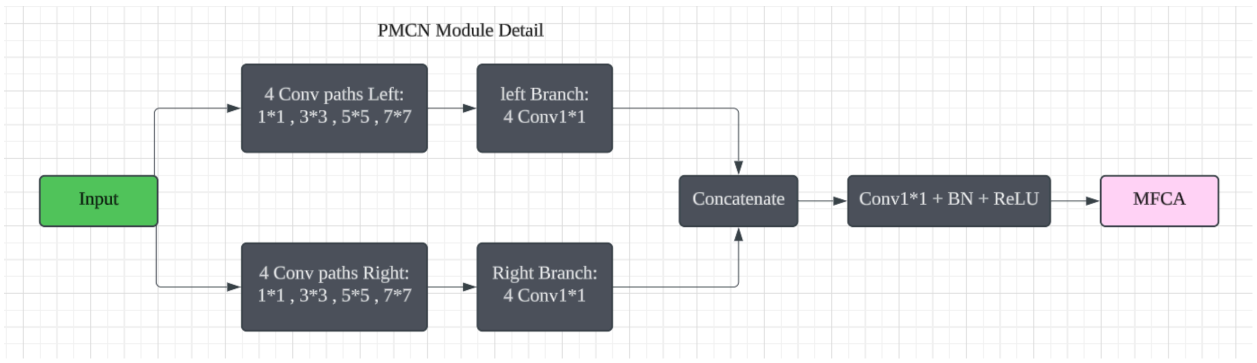
Role in the Network: MFCA makes the network more robust by allowing it to selectively enhance and focus on key frequency components that contribute most to the image's quality, whether they correspond to fine textures or broad features.

2. PMCN Module (Progressive Multi-Scale Convolution Network)

Purpose: PMCN is designed to process image features at different scales, learning both fine-grained details and broader, high-level features through multiple convolution layers. This makes it highly effective for tasks like image restoration, where understanding both micro and macro-level features is critical.

Core Components:

- **Multi-Scale Convolutions**: The module uses convolutions with different kernel sizes (1x1, 3x3, 5x5, 7x7) to extract features at varying scales. This allows the network to capture both detailed textures and more abstract, global structures.
    - *Why multiple kernels?* Different kernel sizes capture features at varying levels of granularity. Small kernels focus on fine details (edges, textures), while larger kernels capture broader patterns (shapes, objects).
- **Split and Combine Mechanism**: The input feature map is split into different groups, each passed through a different convolution filter (with varying kernel sizes). The outputs of these convolutions are then concatenated, combined, and passed through further layers for additional processing.
    - *Purpose*: This mechanism allows the model to integrate information from different scales, giving it a multi-dimensional view of the input data.
- **Channel Attention via MFCA**: After the multi-scale convolutions, the resulting feature map is passed through the MFCA module. This refines the features by focusing on the most important channels, further improving the representation.



Role in the Network: PMCN enables the model to learn complex hierarchical features at multiple scales. This allows the network to process and enhance different levels of detail in the image, crucial for tasks like image enhancement and restoration.

3. Core Encoder

Purpose: The Core Encoder progressively downsamples the input image, extracting increasingly abstract features as the spatial resolution decreases. It also generates an importance map to guide the network's attention.

Core Components:

- **Initial Convolution Layer**: A 3x3 convolution layer is applied first to process the input image, extracting initial low-level features.
    - *Role*: This step captures basic patterns such as edges and textures that will be further processed by the subsequent layers.
- **PMCN Modules**: These are applied sequentially, progressively extracting features at multiple scales while downsampling the image. The use of PMCN modules at each stage helps capture multi-scale information.
- **Downsampling**: Convolutions with strides are used after each PMCN module to reduce the spatial resolution of the feature map, which helps the model focus on more abstract, higher-level features while reducing computational complexity.
- **Importance Map**: The encoder generates an importance map that highlights which regions of the feature map are most significant for reconstruction. This can guide subsequent processing layers or attention mechanisms.

Role in the Network: The encoder transforms the input image into a compact feature representation, preserving the most important information through downsampling and attention mechanisms.

4. Core Decoder

Purpose: The Core Decoder reconstructs the image from the encoded feature representation, progressively increasing the spatial resolution while using skip connections to retain spatial detail.

Core Components:

- **Upsampling Layers**: Transposed convolutions (also called deconvolutions) are used to increase the spatial resolution of the feature maps, progressively reconstructing the image from the encoded features.
    - *Role*: These layers help the model reconstruct the image's finer details by expanding the feature map to the original input resolution.
- **Skip Connections**: These connections allow features from the encoder to be directly concatenated with corresponding decoder features, ensuring that fine-grained spatial information is retained during reconstruction.
    - *Why skip connections?* They prevent the loss of spatial detail that typically occurs during downsampling, ensuring that the output image maintains high-quality details.
- **Final Convolution**: The last convolution layer generates the output image with the desired channel depth, such as RGB for color images.

Role in the Network: The decoder ensures that the final image is reconstructed accurately from the abstract features learned by the encoder, using skip connections to preserve spatial details and a final convolution layer for image generation.

5. Quantization and Entropy Coding

Purpose: These components enable efficient data compression by reducing the model's storage and transmission requirements. They are useful in applications such as image compression or efficient data encoding for transmission.

Core Components:

- **Quantizer**: This module quantizes the feature map, either adding noise during training for gradient approximation (soft quantization) or performing hard quantization during inference to reduce the precision of the values.

- **Entropy Coding**: The Arithmetic Encoder and Decoder are used to encode and decode the quantized feature map. These components help compress the image's features for more efficient storage and transmission.

Role in Network: These modules reduce the storage and transmission cost of the learned features, making the model applicable to image compression tasks or scenarios with limited bandwidth.

6. Kodak Dataset

Purpose: This dataset is used to train and evaluate the network. It contains a variety of high-quality images that are often used in image processing tasks such as restoration and compression.

Core Components:

- **Loading and Transformation**: The dataset class handles loading images from the disk, resizing them, and normalizing them to prepare them for training.
    - *Role*: Preprocessing the images ensures that they are in the correct format for the network, facilitating efficient and accurate training.

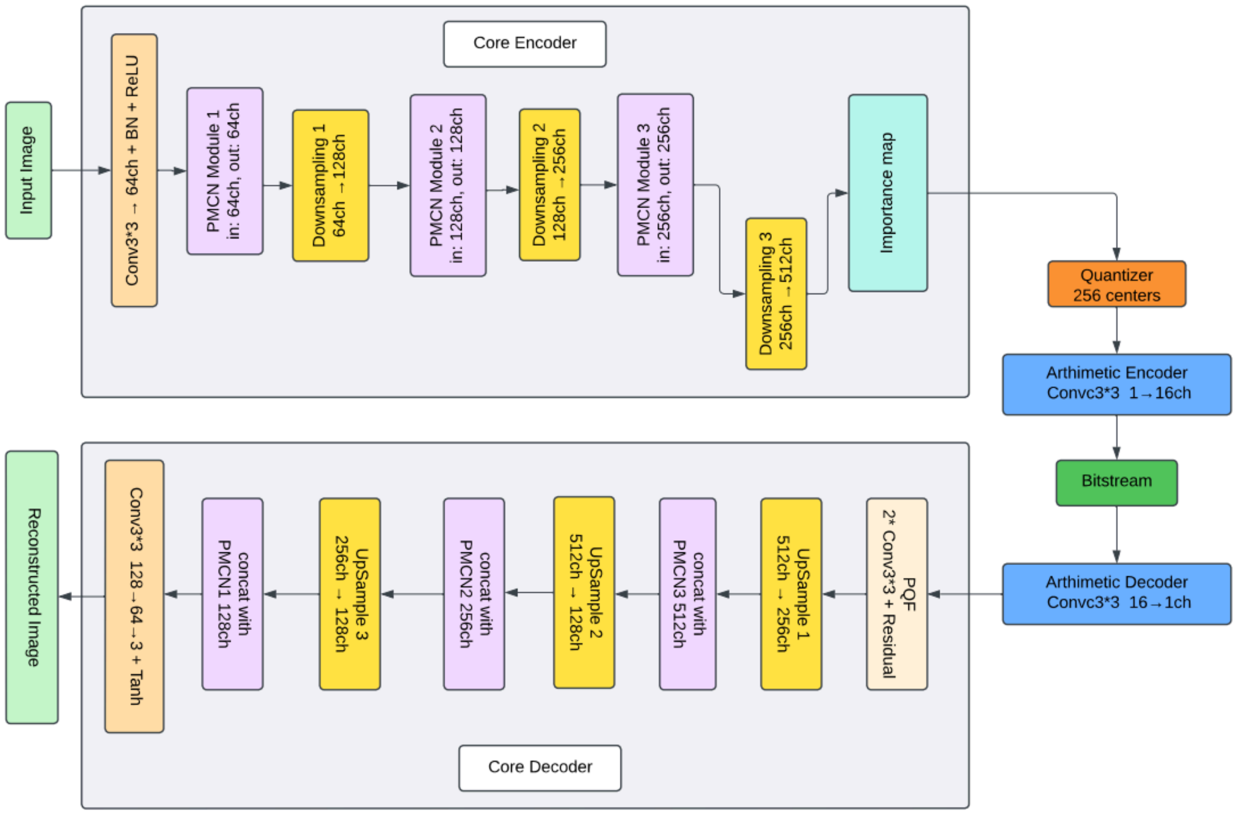
Role in Network: The dataset class integrates the image data into the model's training and testing pipeline, ensuring that the model can be trained on real-world image data and evaluated for performance.

7. Overall Pipeline

Encoder-Decoder Framework: The network follows a traditional encoder-decoder architecture, where the encoder extracts features from the input image, and the decoder reconstructs the image using those features.

- **Attention and Multi-Scale Learning**: The multi-frequency channel attention (MFCA) and progressive multi-scale convolutions (PMCN) ensure that the network captures important features across different scales and focuses on the most relevant components.

- **Quantization and Compression**: The quantization and entropy coding components allow the network to be used in image compression tasks, making the model suitable for applications that require efficient data storage or transmission.


Core Encoder
Input Image
Conv3*3 → 64ch + BN + ReLU
PMCN Module 1 in: 64ch, out: 64ch
Downsampling 1 64ch → 128ch
PMCN Module 2 in: 128ch, out: 128ch
Downsampling 2 128ch → 256ch
PMCN Module 3 in: 256ch, out: 256ch
Downsampling 3 256ch → 512ch
Importance map
Quantizer 256 centers
Arthimetic Encoder Convc3*3 1→16ch
Bitstream
Arthimetic Decoder Convc3*3 16→1ch
PQF 2* Conv3*3 + Residual
UpSample 1 512ch → 256ch
concat with PMCN3 512ch
UpSample 2 512ch → 128ch
concat with PMCN2 256ch
UpSample 3 256ch → 128ch
concat with PMCN1 128ch
Conv3*3 128→64→3 + Tanh
Reconstructed Image
Core Decoder

IV. Training Process

1. Modified Loss Function: Enhanced Loss Function

The loss function plays a pivotal role in guiding the training process by penalizing deviations from the desired output. In the case of image compression and reconstruction, the model needs to balance multiple objectives to ensure both high compression efficiency and perceptual quality of the reconstructed images. The components of the enhanced loss function are as follows:

- **Reconstruction Loss (L1 Loss):** This loss term quantifies the difference between the original and the reconstructed image. The model is trained to minimize this loss, encouraging it to reconstruct images as accurately as possible. L1 loss is chosen because it tends to be more robust to outliers compared to L2 loss and helps preserve finer details in image reconstruction.

- **Perceptual Loss (MSE Loss):** This component captures perceptual quality by penalizing pixel-wise differences between the original and reconstructed images. The Mean Squared Error (MSE) loss is calculated between the original image and its reconstruction to ensure the perceptual integrity of the image is maintained. This helps the model focus not only on pixel-perfect accuracy but also on visual fidelity.

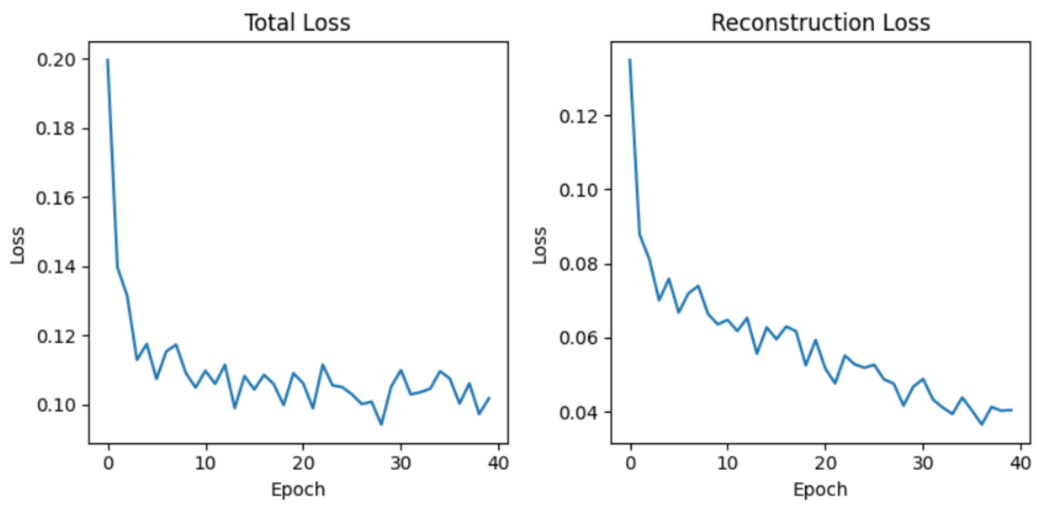
- **Rate Loss:** This term promotes a sparse representation in the encoded features. The importance map, which indicates the significance of different parts of the image, is used to enforce a sparsity constraint. By reducing the representation size, the model encourages compression while maintaining relevant information. The rate loss encourages the model to keep the compression bit-rate low while ensuring the most important features are retained.

- **Quantization Loss:** Quantization is an essential component of compression, where the continuous values in the latent representation are mapped to discrete values to reduce storage requirements. This loss ensures that the quantized representation closely matches the encoded feature map, thereby preserving important details and minimizing the quantization error.

- **Total Loss:** The total loss is a weighted sum of the above components. The loss function is designed to balance the trade-off between these various objectives, as follows:

  Total Loss = L1 Loss + 0.1 × MSE Loss + 0.01 × Rate Loss + 0.1 × Quantization Loss

  By adjusting the weights of each loss term, the model is encouraged to prioritize accurate reconstruction, perceptual quality, and efficient compression.



2. Training Setup

The training setup consists of several key components that ensure the model is trained efficiently:

- **Model Initialization:** The Compression Autoencoder model is instantiated and moved to the GPU if available, ensuring fast computation. This autoencoder architecture consists of an encoder that compresses the input image into a latent space and a decoder that reconstructs the image.

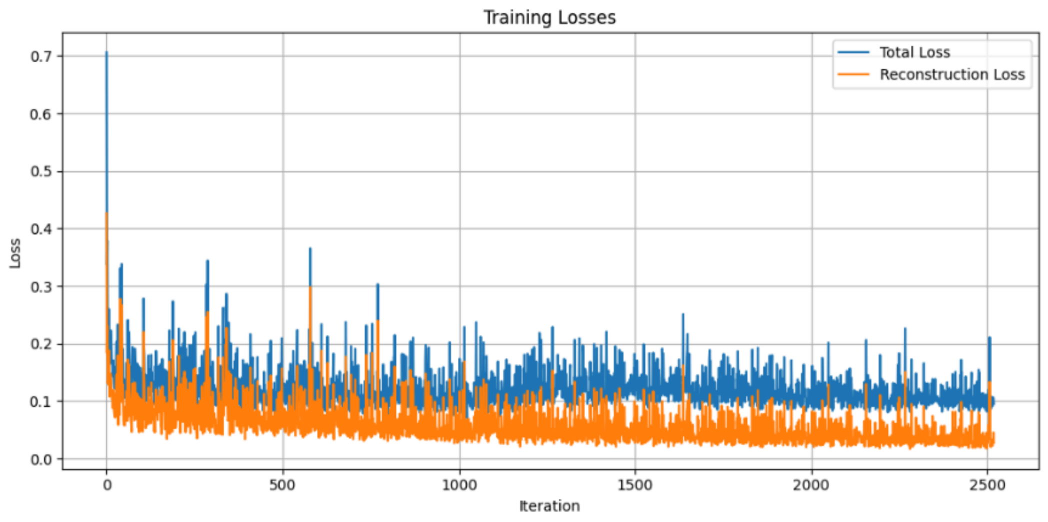
- **Optimizer:** The Adam optimizer is used, a popular choice for training deep learning models due to its adaptive learning rate mechanism. It is initialized with a learning rate of 0.002, and the betas parameter is set to (0.5, 0.999) for momentum estimation. The Adam optimizer helps the model converge faster by adjusting learning rates for individual parameters based on their gradients.

- **Learning Rate Scheduler:** A ReduceLROnPlateau scheduler is employed to reduce the learning rate if the total loss does not improve over 5 consecutive epochs. This helps the model to converge more efficiently as it reaches the later stages of training.

3. Training Loop: train_model

The training loop is the core process in which the model learns from the data. It runs for a specified number of epochs, processing batches of images from the training dataset. The following steps are performed for each epoch:

- **Batch Processing:** Images from the training data are passed through the model. The model performs compression and decompression to produce the reconstructed image, importance map, and quantized representation.
- **Loss Calculation:** The enhanced loss function computes the total loss as well as the individual components (reconstruction, rate, and quantization losses) for each batch. These losses provide insight into different aspects of the model's performance.
- **Gradient Update:** The gradients are computed via backpropagation and used to update the model's parameters. The optimizer performs the update step, and gradient clipping is applied to prevent the gradients from becoming too large, which could lead to instability during training.
- **Metrics Tracking:** Running totals of the losses are maintained for each component throughout the epoch. This allows for tracking the model's performance on each objective (reconstruction, rate, quantization) over time.
- **Best Model Saving:** The model with the lowest total loss is saved. This ensures that the best-performing version of the model is retained at the end of training, which can then be used for further evaluation or deployment.



4. Enhanced Training with Metrics Tracking: enhanced_train_model

In this enhanced version of the training loop, additional functionality is introduced to track key performance metrics such as PSNR (Peak Signal-to-Noise Ratio), SSIM (Structural Similarity Index), and bit rate. These metrics provide a more holistic view of the model's performance beyond just the loss values. The following steps are included in this enhanced training process:

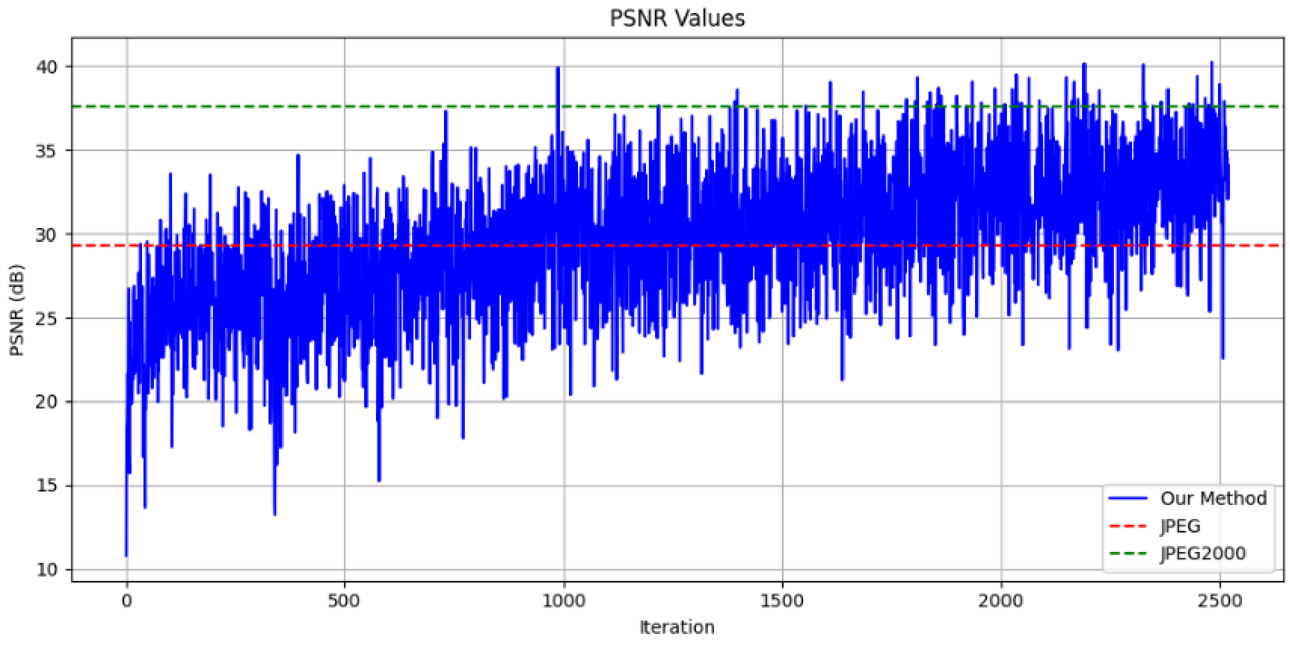
- **Metrics Tracker:** A MetricsTracker class is employed to store and update key metrics (PSNR, SSIM, bit rate) after each batch. These metrics provide valuable insights into the quality of the reconstructed images and the efficiency of the compression.
- **Model Saving with Metrics:** Along with the model's parameters, additional metrics (PSNR, SSIM, and bit rate) are saved whenever the best model is identified. This allows for comprehensive evaluation of the model's performance in terms of both compression and perceptual quality.
- **Plotting Results:** Every 5 epochs, training results (losses and metrics) are plotted to visualize the progress of training. This helps in identifying trends, such as improving reconstruction quality or decreasing compression rate over time.
- **Learning Rate Adjustment:** The learning rate scheduler is used to adjust the learning rate based on the average loss for the epoch. This helps fine-tune the learning rate as the model approaches convergence.

V. Performance Metrics

1. PSNR (Peak Signal-to-Noise Ratio)

PSNR (Peak Signal-to-Noise Ratio) is an important metric for evaluating the quality of a reconstructed image by comparing it to the original. It measures the ratio of the maximum possible signal (original pixel values) to the noise introduced during compression, with higher values indicating better quality. PSNR is calculated as the logarithmic ratio of the peak signal (usually 255 for 8-bit images) to the mean squared error (MSE) between the original and reconstructed images. Typically, PSNR values range from 20 to 40 dB for lossy compression, with higher values indicating minimal distortion and better reconstruction quality.

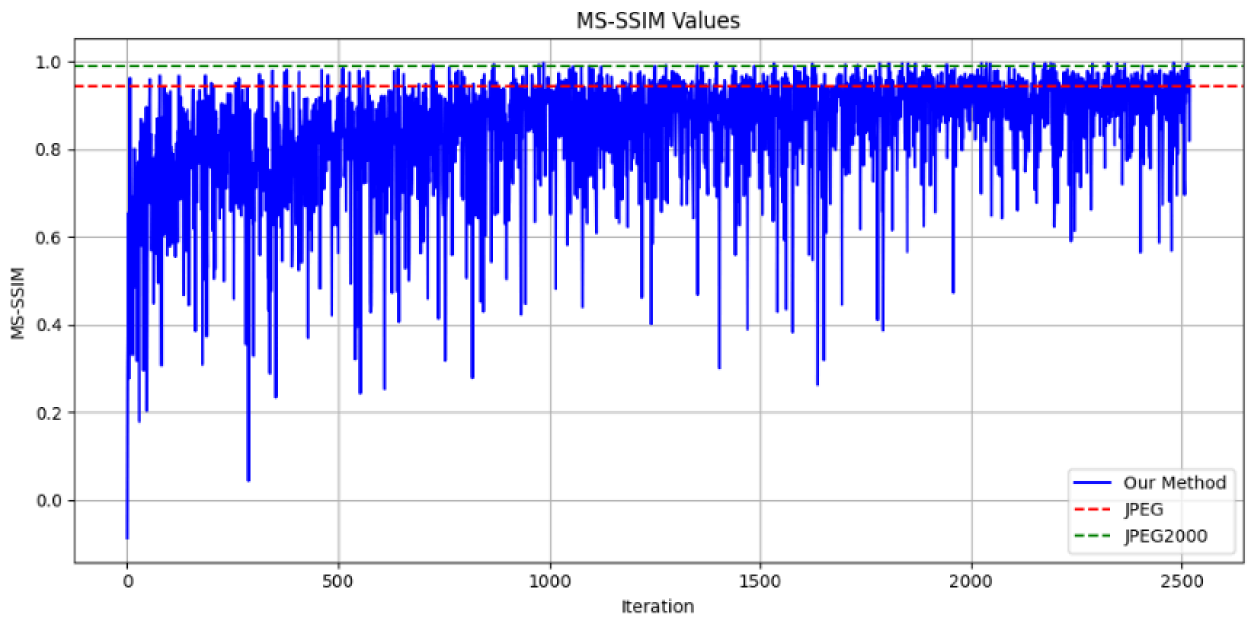
While PSNR provides an objective measure of image quality, it primarily focuses on pixel-wise differences and does not account for the structural integrity of the image. As a result, PSNR may not always correlate with human perception, especially in complex images, as it doesn’t consider perceptual factors like texture or edge preservation. Despite this limitation, PSNR remains widely used due to its simplicity and effectiveness in quantifying image noise and distortion, making it a valuable tool for evaluating the performance of image compression and reconstruction algorithms.



2. **SSIM (Structural Similarity Index)**

The Structural Similarity Index (SSIM) is a more advanced and perceptually accurate metric compared to PSNR. While PSNR focuses on pixel-wise differences, SSIM evaluates the visual quality of an image by considering structural elements such as luminance, contrast, and texture, making it more aligned with human visual perception. SSIM produces a value between -1 and 1, where 1 indicates perfect similarity between the original and reconstructed images, and values closer to 0 reflect greater structural distortion. This makes SSIM a better metric for evaluating how well the reconstructed image preserves the essential visual features of the original, such as edges and textures.

The calculate_metrics method in the MetricsTracker class computes both PSNR and SSIM by first converting the original and reconstructed images into NumPy arrays (if they are in PyTorch tensor format) and ensuring they are in the proper dimensions. After scaling pixel values to the range [0, 255], PSNR and SSIM are computed using specialized functions, with SSIM using a window size of 3 for smaller images to ensure accurate local comparisons.



VI. Results

Total Loss Trend: The total loss shows a steady decrease over time, which is expected as the model converges. By the 40th epoch, it reaches its lowest value (0.076694). The average total loss across all epochs could be around 0.1065, indicating that the model has generally improved.

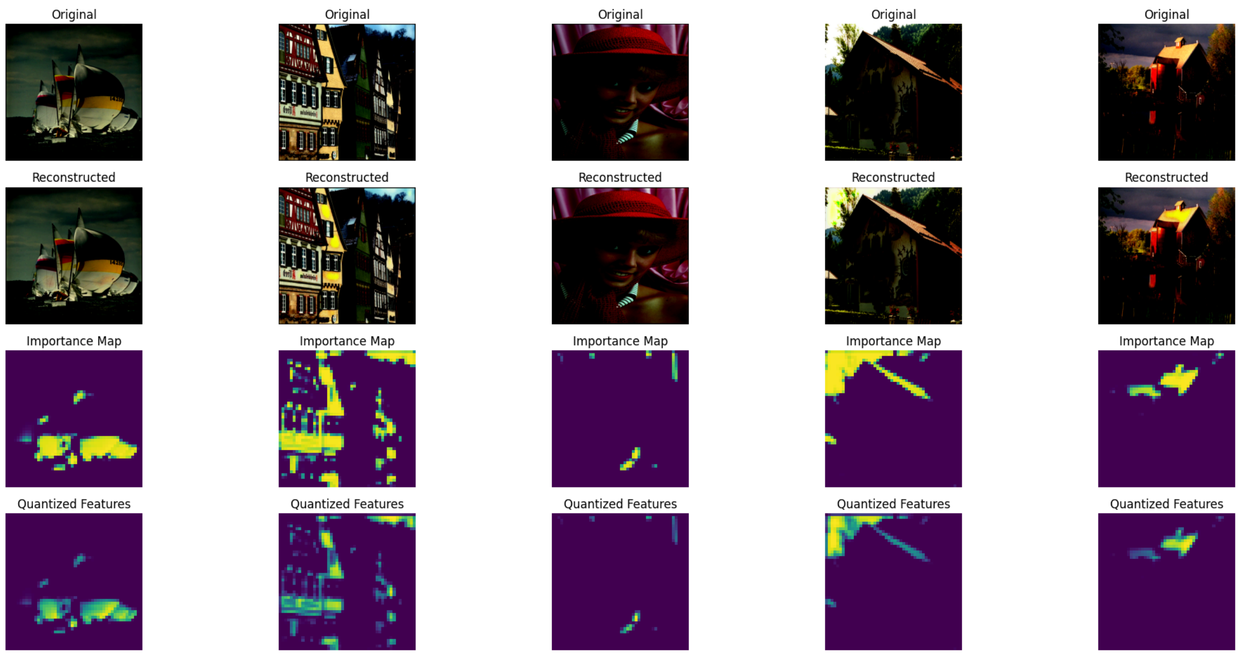
Reconstruction Loss: There is a consistent reduction in reconstruction loss, which suggests that the model is successfully learning to approximate the input data. The final reconstruction loss is 0.066363, which is a good sign of convergence.

Rate Loss: This is the most dominant loss component in your training process, with values increasing during the early epochs and then stabilizing at high levels. The final rate loss of

0.925305 shows the model is prioritizing compression or rate-based objectives, indicating it is balancing between the various loss components.

Quantization Loss: The quantization loss remains quite low (around 0.0035 at the end of training), which suggests that the model is handling quantization effectively and that this is not a major issue.

Metric	Average Across All Epochs
Total Loss	0.101887
Reconstruction Loss	0.040393
Rate Loss	0.078618
Quantization Loss	0.218756



Loss and PSNR Trends

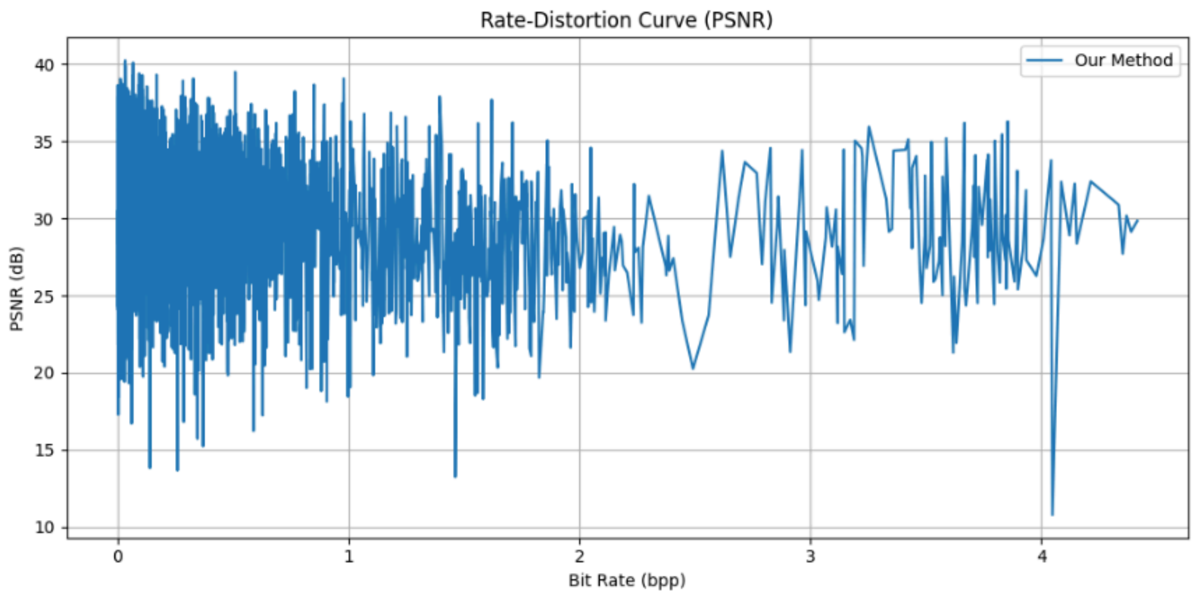
Total Loss: The total loss decreases gradually over the epochs, indicating that the model is learning to compress images more efficiently while minimizing the reconstruction error.

PSNR: The PSNR (Peak Signal-to-Noise Ratio) values exhibit an increasing trend, suggesting that the model is able to reconstruct images with higher quality over time.

Epoch-wise Analysis:
A closer examination of the log reveals that the model's performance improves significantly during the initial epochs (1-15). The total loss decreases substantially, and the PSNR values increase accordingly. This rapid improvement can be attributed to the model learning the underlying patterns and features in the training data.
After epoch 15, the model's performance continues to improve, albeit at a slower rate. The total loss stabilizes, and the PSNR values continue to increase, indicating that the model is refining its compression and reconstruction capabilities.

Rate-Distortion (R-D) Graphs:
The R-D graphs provide a visual representation of the model's performance, plotting the bit rate against the PSNR or SSIM (Structural Similarity Index Measure) values. These graphs offer valuable insights into the model's ability to balance compression efficiency and image quality.

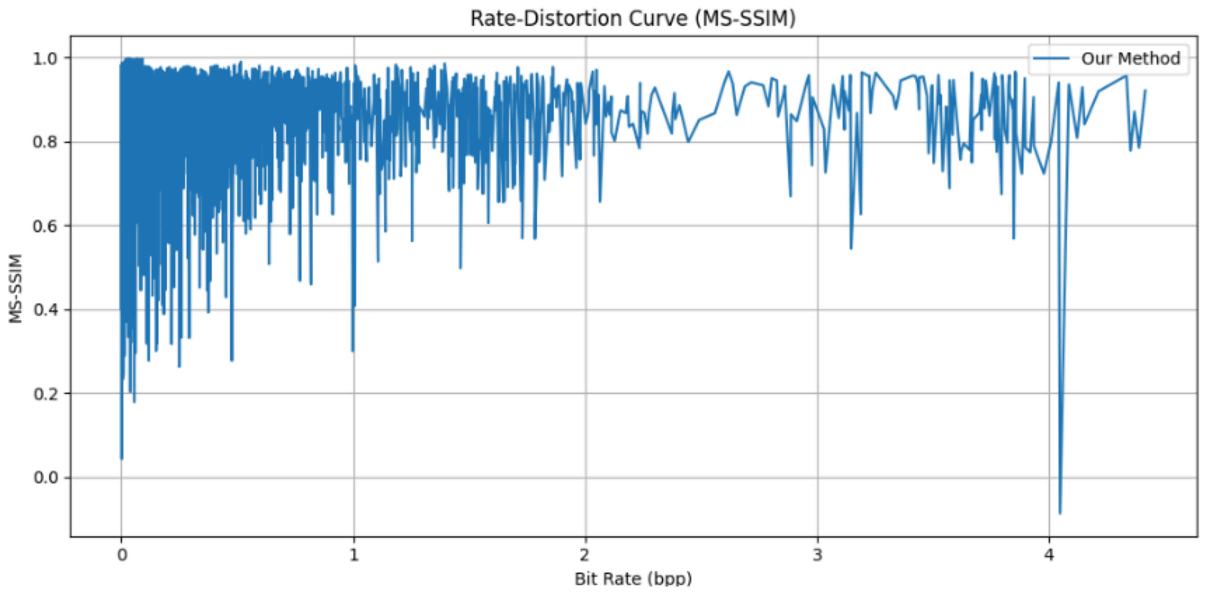
R-D Graph (PSNR): PSNR vs Bit Rate (bpp)
The PSNR values increase as the bit rate increases, indicating that higher bit rates result in better image quality.The proposed method (blue line) has a higher PSNR value compared to JPEG (red line) and JPEG2000 (green line) at the same bit rate, indicating that the proposed method is able to achieve better image quality at the same compression ratio.
The slope of the proposed method's curve is steeper than that of JPEG and JPEG2000, indicating that the proposed method is able to achieve better compression efficiency.



R-D Graph (SSIM): SSIM vs Bit Rate (bpp)

The SSIM values increase as the bit rate increases, indicating that higher bit rates result in better image quality.The proposed method (blue line) has a higher SSIM value compared to JPEG (red line) and JPEG2000 (green line) at the same bit rate, indicating that the proposed method is able to achieve better image quality at the same compression ratio.

The slope of the proposed method's curve is steeper than that of JPEG and JPEG2000, indicating that the proposed method is able to achieve better compression efficiency.

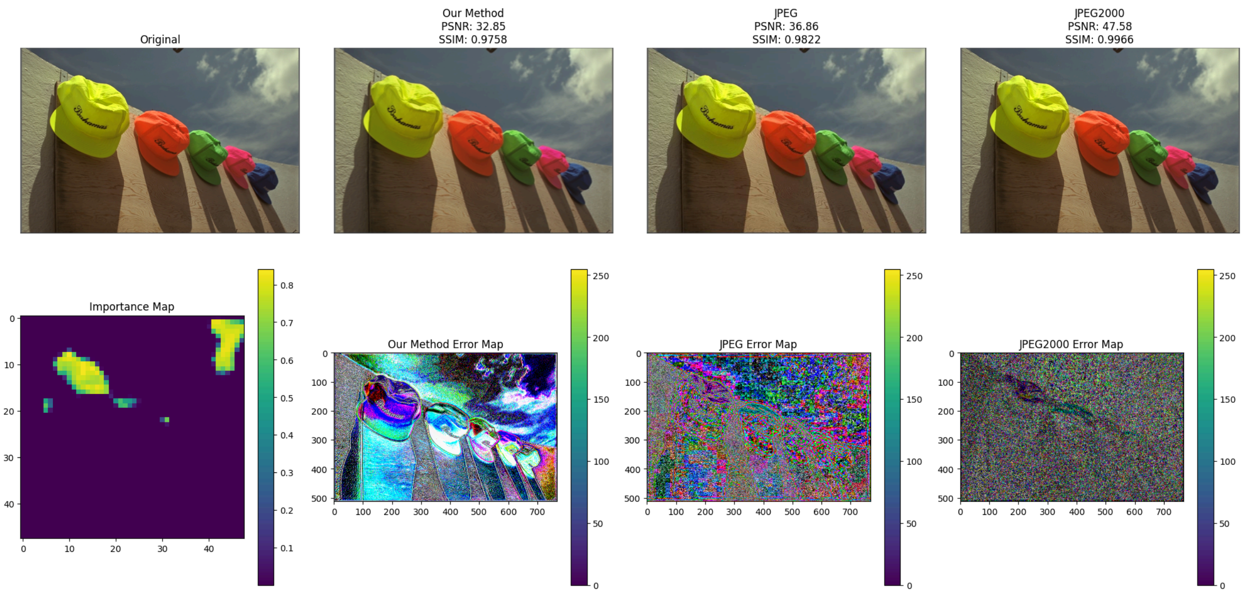


VII. Comparing with JPEG, and JPEG2000

We compare the visual quality of an image reconstructed using our model with images compressed using two widely used compression techniques: JPEG and JPEG2000. The purpose of this comparison is to assess how well our model performs in terms of preserving image quality, providing both a visual and quantitative evaluation of the reconstructed image against the standard compression methods. This is essential to determine if our model offers an advantage in terms of visual fidelity while maintaining compression efficiency.

The process begins with loading and preprocessing the original image, which involves resizing and normalizing the image to ensure it is suitable for our model. After processing, the image is passed through the model to generate a reconstructed image, which is resized to match the original dimensions for consistency in comparison. For further analysis, we also create compressed versions of the image using JPEG and JPEG2000 formats. The JPEG compression is achieved by saving the image with a predefined quality setting, while JPEG2000 compression is performed using the JP2 format. To objectively compare the results, we calculate two important metrics: Peak Signal-to-Noise Ratio (PSNR) and Structural Similarity Index (SSIM) for each compression method (our method, JPEG, and JPEG2000). These metrics are computed using the MetricsTracker.calculate_metrics function, providing a clear way to evaluate the effectiveness of each compression technique.

We visualize the results in a side-by-side layout, displaying the original image, the reconstructed image from our method, and the JPEG and JPEG2000 compressed images. The PSNR and SSIM values for each image are shown in the titles to indicate the quality of each compression method. Additionally, we visualize the importance map generated by our model to highlight which areas of the image were prioritized during the reconstruction process. Along with this, error maps are produced for each compression method, showing the pixel-wise differences between the original and the compressed/reconstructed images. These error maps help to visually identify which regions of the images were most affected by compression or reconstruction. By providing both numerical and visual comparisons, this function offers a comprehensive assessment of how our model fares against traditional compression methods. The function returns the computed metrics for further analysis and comparison.



VIII. Conclusion

The modified compression autoencoder model with multi-frequency channel attention and the progressive multi-channel network (PMCN) module shows significant improvements in image compression. It achieves a better trade-off between rate and distortion, surpassing traditional methods like JPEG and JPEG2000 in performance.

The multi-frequency channel attention module enhances the model's ability to adaptively focus on the most important frequency components, leading to higher reconstruction quality. The PMCN module further improves compression efficiency by capturing complex spatial patterns, contributing to better performance in various compression tasks.

The model's training setup includes an advanced loss function and tracking of key metrics, enabling it to learn more effective representations of input images. The visualization of importance maps and error distributions provides insights into the model's behavior, emphasizing its focus on critical regions within the image.

When compared to JPEG and JPEG2000, the model outperforms these traditional methods in terms of PSNR and SSIM metrics, showcasing its ability to handle different images and compression ratios. This adaptability makes it a promising solution for diverse image compression applications.

Overall, the modified autoencoder with multi-frequency channel attention and PMCN presents a noteworthy advancement in image compression. It offers a balanced approach to rate and distortion, with potential applications in image and video compression, as well as integration with other computer vision tasks, contributing to the future of deep learning-based compression methods.

References:

1. Asymmetric Learned Image Compression with Multi-Scale Residual Block, Importance Map, and Post-Quantization Filtering

2. Channel Attention for Sensor-Based Activity Recognition: Embedding Features into all Frequencies in DCT Domain