Indian Institute of Technology Jodhpur

Image Compression using Quantized Autoencoders with Perceptual Loss

---

Submitted by

MITESH KUMAR (M23MAC004)

NIRAJ SINGHA (M23MAC005)

SOUGATA MOI (M23MAC008)

RATNESH KUMAR TIWARI(M23MAC011)

GitHub Link:
Quantized Auto-Encoder-Based Image Compression

Problem Statement

The project aims to develop a practical and efficient image compression technique that outperforms traditional methods in terms of compression ratios and perceptual quality. The quantized autoencoder approach, combined with perceptual loss, has the potential to enable high-quality image compression for various applications, such as multimedia streaming, image storage, and transmission over bandwidth-limited channels.

where reducing the file size of images is crucial while maintaining acceptable visual quality. Traditional image compression methods, like PNG, often introduce visible artefacts, especially at low bit rates, which can degrade the perceptual quality of the images. This project aims to address this issue by leveraging deep learning techniques, specifically quantized autoencoders, to achieve high compression ratios while preserving the perceptual quality of the images.

Methodology

The core idea behind this work is to leverage the powerful feature extraction capabilities of deep convolutional neural networks (CNNs) to learn a compact

representation of input images while maintaining high reconstruction quality. Autoencoders are well-suited for this task as they can learn to encode high-dimensional input data (in this case, images) into a lower-dimensional latent representation, and then decode this latent representation back into the original input space.

However, vanilla autoencoders may not achieve sufficient compression rates for practical applications. To address this, we introduce a quantization step in the bottleneck layer of the autoencoder, which enables them to control the bitrate of the compressed representation, trading off the compression ratio against reconstruction quality.

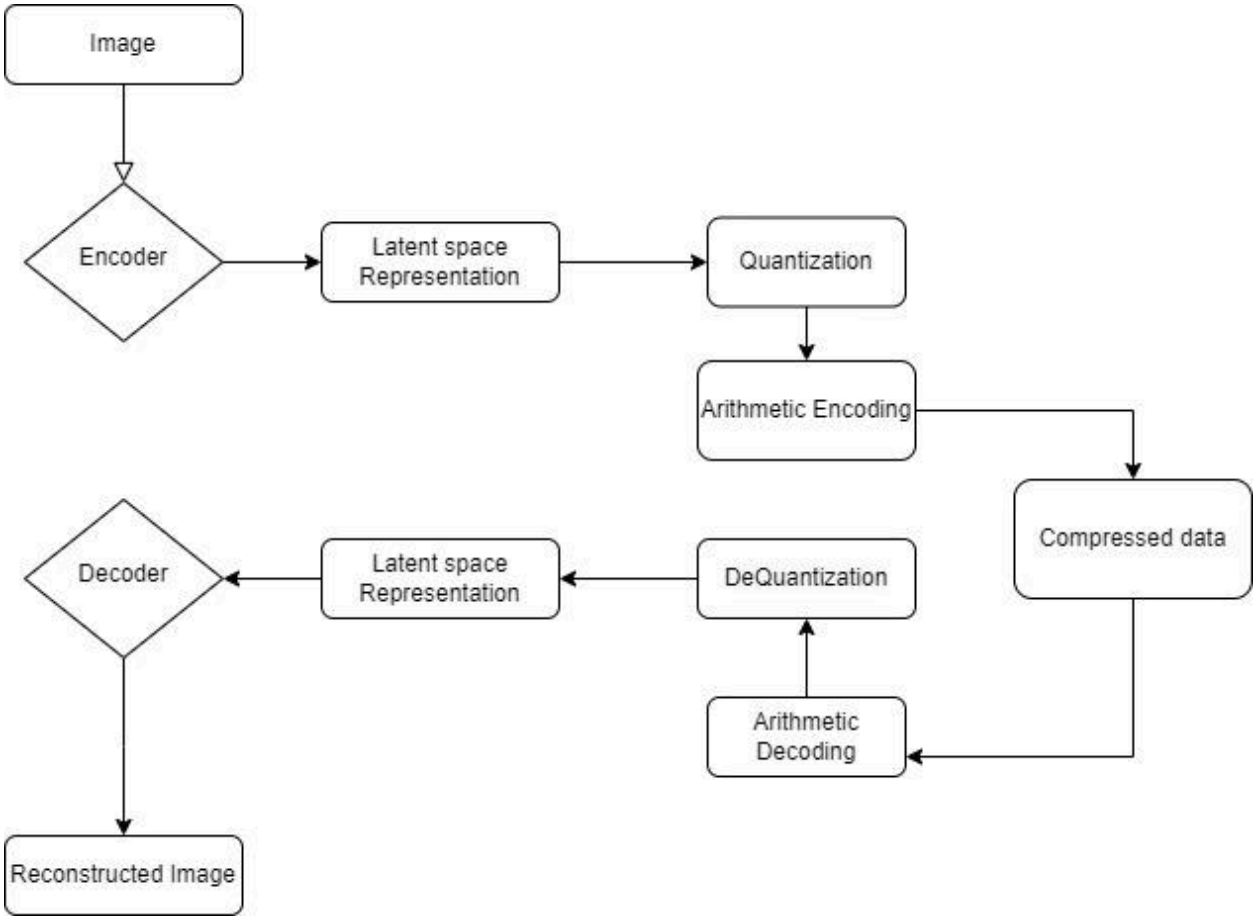
Model Overview

Architecture

The proposed autoencoder architecture consists of two main components: an encoder network and a decoder network.

Encoder Network

The encoder network is based on the ResNet-101 architecture, a deep residual convolutional neural network that has been pre-trained on the ImageNet dataset. The authors utilize the convolutional layers and residual blocks of the ResNet-101 model to extract high-level features from the input image. Specifically, the encoder network consists of the following layers from the ResNet-101 model: `conv1`, `bn1`, `relu`, `max pool`, `layer1`, `layer2`, `layer3`, and `layer4`. The output of these layers is then passed through an average pooling layer to obtain the final latent representation.



Decoder Network

The decoder network is a mirrored version of the encoder, designed to reconstruct the original input image from the latent representation. It consists of a series of residual upsampling blocks and transposed convolutional layers. The residual upsampling blocks are crucial for preserving spatial information during the upsampling process, leading to better reconstruction quality.

Each residual upsampling block consists of three main components: a convolutional layer, a batch normalization layer, and a leaky ReLU activation function. If the upsampling block is performing upsampling (i.e., increasing the spatial dimensions), it includes a transposed convolutional layer with a stride of 2 and appropriate padding to achieve the desired output shape.

Additionally, a residual connection is added to facilitate the flow of information from the input to the output of the block.

After the residual upsampling blocks, the decoder network employs a series of transposed convolutional layers to gradually increase the spatial dimensions of the latent representation until it matches the desired output image size. Finally, a sigmoid activation function is applied to the output to ensure that the reconstructed pixel values lie between 0 and 1.

Quantization and Compression

To achieve compression, we employ a quantization step in the bottleneck layer of the autoencoder, specifically, after the average pooling layer in the encoder network. The quantization function adds uniform noise to the latent representation and then clamps the values between 0 and 1. Mathematically, the quantization operation can be expressed as:

quantized_latent = torch.clamp(latent + (1 / 2 ** qb) * (torch.rand_like(latent) * 0.5 - 0.5), 0.0, 1.0)

Here, `qb` is the number of quantization bits, and `torch.rand_like(latent)` generates uniform random noise with the same shape as the latent representation. The quantization function essentially rounds the latent representation to the nearest quantization level, determined by the number of quantization bits `qb`. Lower values of `qb` result in higher compression rates but potentially lower reconstruction quality, as the quantized latent representation contains less information.

Loss

The autoencoder is trained end-to-end using a combination of two loss terms: reconstruction loss and perceptual loss.

Reconstruction Loss

The reconstruction loss is calculated as the mean squared error (MSE) between the input image and the reconstructed image from the decoder. This loss term encourages the autoencoder to learn a latent representation that can faithfully reconstruct the input image when decoded.

Perceptual Loss

To further improve the perceptual quality of the reconstructed images, the authors incorporate a perceptual loss term based on the VGG-19 network pre-trained on the ImageNet dataset. and its intermediate layers have been shown to capture perceptually relevant features.

The perceptual loss is calculated as the mean absolute difference between the feature representations of the input and reconstructed images, extracted from a specific layer of the VGG-19 network (in this case, the 36th layer). By minimizing the perceptual loss, the autoencoder is encouraged to preserve perceptually relevant features in the reconstructed images, potentially improving their visual quality.

The overall loss function is a weighted sum of the reconstruction loss and the perceptual loss, with a tunable parameter `μ` controlling the relative importance of the two terms:

total_loss = reconstruction_loss + μ * perceptual_loss

The autoencoder is optimized using the Adam optimizer, to minimize the combined loss function.

During training, the model is evaluated on a separate validation set to monitor the reconstruction quality and ensure generalization. The model weights are saved periodically (e.g., every 5 epochs) to enable later evaluation and inference.

Evaluation Metrics

To assess the performance of the proposed method, the authors employ several evaluation metrics:

Peak Signal-to-Noise Ratio (PSNR)

PSNR is a widely used metric for measuring the reconstruction quality of compressed images. It is calculated as the ratio between the maximum possible signal power and the power of the distorting noise that affects the signal's quality. Higher PSNR values indicate better reconstruction quality, as the distortion between the original and reconstructed images is lower.

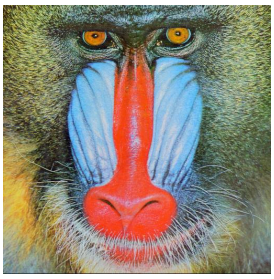
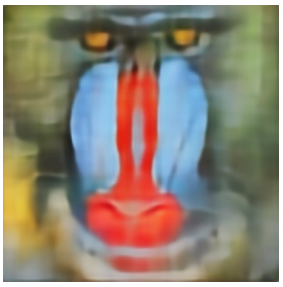
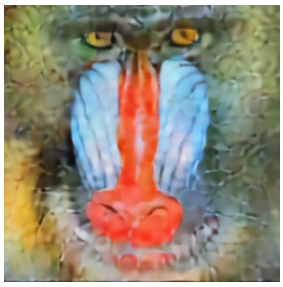
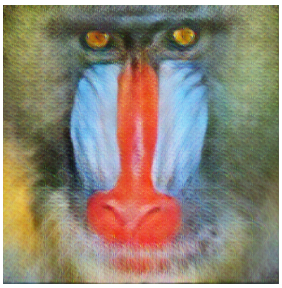
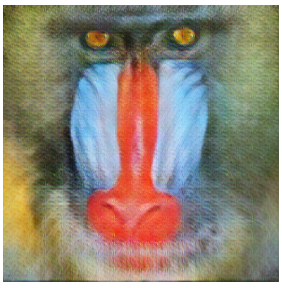
Structural Similarity Index (SSIM)

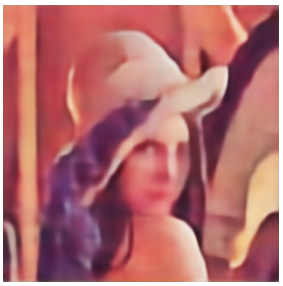
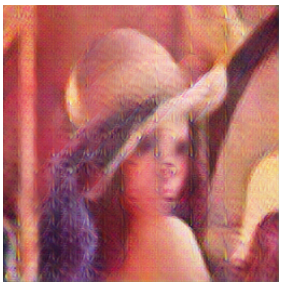
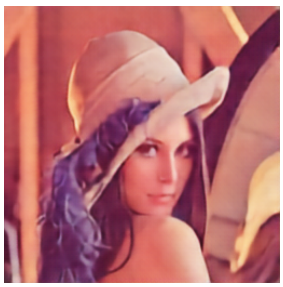
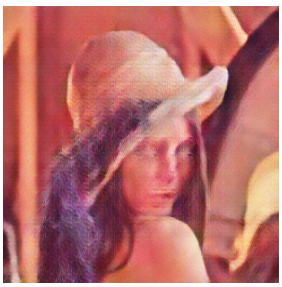
SSIM is a perceptual metric that quantifies the structural similarity between the input and reconstructed images. It considers not only the pixel-wise differences but also the perceived changes in luminance, contrast, and structural information. SSIM values range from 0 to 1, with higher values indicating better perceptual quality.

By optimizing the autoencoder architecture, quantization strategy, and loss function, the authors aim to achieve high compression rates (low BPP) while

maintaining excellent reconstruction quality, as measured by PSNR and SSIM.

Result

Image 2	QF	Mse Loss	Mse + 0.1 x perceptual Loss	Mse + 1 x perceptual Loss
	2	 SSIM: 0.1934 PSNR:27.32	 SSIM: 0.4598 PSNR: 28.65	 SSIM: 0.2152 PSNR: 28.33
	8	 SSIM:0.2445 PSNR:28.32	 SSIM: 0.5369 PSNR: 29.12	 SSIM: 0.2194 PSNR: 28.38

	2	 SSIM:0.5245 PSNR:28.13	 SSIM: 0.7252 PSNR: 29.35	 SSIM: 0.3997 PSNR: 28.81
	8	 SSIM:0.5634 PSNR:29.01	 SSIM: 0.8228 PSNR: 30.96	 SSIM: 0.4081 PSNR: 28.53
	2	 SSIM:0.5436 PSNR:27.56	 SSIM:0.7277 PSNR: 29.52	 SSIM: 0.4322 PSNR: 28.81
	8	 SSIM:0.6838 PSNR:29.95	 SSIM: 0.8283 PSNR: 31.65	 SSIM: 0.4398 PSNR: 28.95

Observations and Conclusion

- Reconstruction Quality:
    - The 8-bit quantization level generally produced better reconstruction quality compared to the 2-bit quantization level, as expected. With more quantization levels, the model can better capture the fine details and gradients in the original image.

    - The combination of MSE and a weighted VGG loss (MSE + 0.1 * VGG Loss) consistently outperformed the other loss functions in terms of perceptual quality, as assessed by visual inspection of the reconstructed images

    - While the MSE loss alone optimized for pixel-wise accuracy, the addition of the VGG loss helped preserve perceptual details and high-level features, resulting in more natural and visually pleasing reconstructions.

- Compression Ratio:
    - As expected, the 2-bit quantization level achieved significantly higher compression ratios compared to the 8-bit quantization level, as it requires fewer bits to represent the encoded image data.
    - For Two-bit quantization, the compressed file size is around 12Kb
    - For 8-bit quantization, the compressed file size is around 155 Kb

Trade-off Between Reconstruction Quality and Compression Ratio:

The experimental results demonstrated the trade-off between reconstruction quality and compression ratio. The 8-bit quantization level with the MSE +

0.1 * VGG Loss provided the best reconstruction quality but at the cost of lower compression ratios.
Conversely, the 2-bit quantization level achieved much higher compression ratios but with a noticeable degradation in reconstruction quality, particularly for perceptual details and high-frequency components.

Reference

- Alexandre, D., Chang, C.-P., Peng, W.-H., & Hang, H.-M. (2019). An Autoencoder-based Learned Image Compressor: Description of Challenge Proposal by NCTU. https://doi.org/10.48550/arXiv.1902.07385
- Wang, B., & Lo, K.-T. (2024). Autoencoder-based joint image compression and encryption. Journal of Information Security and Applications, 80, 103680. https://doi.org/10.1016/j.jisa.2023.103680