Image Compression and Performance Evaluation

***Abstract*—Image compression is a very interesting and important task in linear algebra. In this paper, we aim to evaluate some popular and advanced image compression methods. We use classical mathematical metrics for evaluation, and we also want to understand how this will affect computer vision networks.**

***Index Terms*—compression, computer vision, SVD, linear algebra**

I. Introduction

Image compression is a very interesting and important task for linear algebra, in this paper, we want to try and evaluate some popular and advanced image compression methods. We evaluate with classical math metrics, also we want to know how this will affect to computer vision networks. The source code and experimental results used in this study have been made publicly available at: https://github.com/HungryNeko/EE510project

II. Background

Images are used in many fields such as medicine to terrain mapping. There is a need to compress the images and represent them in shorter form for effective transmission. Numerous techniques are present for compressing image data. The main types can be broadly classified into two types; lossy and lossless. For lossless images domains such as military, medical image printing, and land terrain demand that the entire image with complete quality be present. Other lossy techniques like JPEG, wavelet transform are capable to do high compression ratio tasks.

III. Methodology

A. Compression Methods

1) JPEG (Lossy Compression): The Joint Photographic Experts Group (JPEG) standard constitutes the predominant mechanism for the compression of continuous-tone photographic imagery. Its fundamental methodology is predicated on the Discrete Cosine Transform (DCT). The algorithm leverages the psychovisual properties of the human visual system, which exhibits significantly higher sensitivity to luminance variations than to high-frequency chromatic fluctuations.

The compression process partitions the image into 8×8 blocks and applies a forward 2D DCT to transform spatial domain data into the frequency domain. Compression is subsequently achieved through the quantization of DCT coefficients, wherein high-frequency components are selectively attenuated or discarded to reduce data volume.

2) PNG (Lossless Compression): Portable Network Graphics (PNG) serves as the de facto standard for lossless image compression. It employs the Deflate algorithm, a hybrid implementation of LZ77 sliding-window compression and Huffman entropy coding. Unlike lossy counterparts, PNG operates by detecting and encoding redundant patterns within the data stream, thereby preserving absolute data integrity. This ensures a bit-exact reconstruction of the original image and supports alpha-channel transparency. However, for photographic imagery characterized by high spectral variance, PNG generally results in larger file sizes compared to lossy techniques.

3) K-means Clustering (Color Quantization): While conventionally employed for unsupervised data classification, K-means clustering is implemented herein for Color Quantization (Vector Quantization). The algorithm partitions the image's chromatic space into K distinct clusters, deriving K representative centroids to approximate the original color distribution.

The objective is to minimize the Within-Cluster Sum of Squares (WCSS). Given a set of pixels $\{x_1, ..., x_n\}$ and clusters $S = \{S_1, ..., S_K\}$, the optimization problem is defined as:

$$J = \sum_{i=1}^{K} \sum_{x_j \in S_i} ||x_j - \mu_i||^2 \tag{1}$$

where μ_i denotes the centroid of cluster S_i. This process yields a substantial reduction in bit-depth by mapping the continuous color space to a discrete, limited palette (e.g., $K = 16$).

4) Principal Component Analysis (PCA): Principal Component Analysis (PCA) is a statistical technique utilized for dimensionality reduction. In the context of image compression, PCA operates on the principle of orthogonal transformation to extract vectors representing the directions of maximal variance (information) within the image matrix.

For a data matrix X (where rows represent pixel vectors), PCA computes the covariance matrix C:

$$C = \frac{1}{n-1} X^T X \tag{2}$$

The principal components are defined as the eigenvectors v derived from the eigendecomposition:

$$Cv = \lambda v \tag{3}$$

where λ represents the eigenvalues. Dimensionality reduction is realized by projecting the image data onto the subspace spanned by the top k principal components associated with the largest eigenvalues, effectively compressing the data while simultaneously attenuating noise.

5) SVD: The SVD uses the number of singular values to compress the image. In SVD computation, the original image matrix A can be represented as:

$$A = U\Sigma V^T \tag{4}$$

If we only consider the first k singular values, the image array A can be represented with a similar matrix A'. In this case, the number of singular values k is used to control the compression ratio and the compression quality. With larger k, the SVD algorithm may catch more information from the original image array, thus producing better results.

6) PatchSVD: SVD can be efficient and acceptable for compressing simple images. However, SVD may struggle when handling complex images or when a high compression ratio is required. Besides, the features in the image are not evenly distributed, thus the SVD may not be able to collect enough features. To solve this problem, there is an improved algorithm called PatchSVD. The PatchSVD first computes the SVD of the image with the first k singular values. Then, it subtracts the k-SVD approximation from the original image and gets the difference matrix δ. The SVD can catch most of the information in the image, and the δ indicates the complex component in the image. The image is split into small patches to distinguish the complex parts and simple parts. Next, a score function using standard deviation was introduced to classify each patch. In the end, PatchSVD uses a small k value k-simple and a large k value k-complex to compute SVD for corresponding patches and assemble them together to form a complete compressed image.

B. Evaluation Methods

For evaluation, we use classical mathematical methods to assess compression quality, and image classification networks to mimic the overall discernibility perceived by the human eye.

1) SSIM: The structural similarity index measure(SSIM) measures the similarity of two images based on luminance, contrast, structure. The basic idea of SSIM is that people care more about structural information instead of pixel difference when comparing images [1].

$$SSIM(x,y) = \frac{(2\mu_x\mu_y + C_1)(2\sigma_{xy} + C_2)}{(\mu_x^2 + \mu_y^2 + C_1)(\sigma_x^2 + \sigma_y^2 + C_2)} \tag{5}$$

where:

- μ_x, μ_y: mean intensity of images x and y
- σ_x^2, σ_y^2: variance of images x and y
- σ_{xy}: covariance between x and y
- C_1, C_2: constants to stabilize division (avoid denominator = 0)

2) MSE: The mean squared error (MSE) is a classical metric that measures the average squared difference between the original image and the reconstructed image. Only pixel-wise fidelity is evaluated without considering perceptual aspects [2].

$$MSE = \frac{1}{MN}\sum_{i=0}^{M-1}\sum_{j=0}^{N-1}\left(I(i,j) - K(i,j)\right)^2 \tag{6}$$

where:

- M, N: width and height of the image
- $I(i,j)$: pixel value of the original image at position (i,j)
- $K(i,j)$: pixel value of the reconstructed image at position (i,j)

3) PSNR: The peak signal-to-noise ratio(PSNR) is commonly used to measure the quality of image reconstruction, normally defined by MSE [3].

$$PSNR = 10 \cdot \log_{10}\left(\frac{MAX_I^2}{MSE}\right) \tag{7}$$

where:

- MAX_I: maximum possible pixel value of the image (e.g., 255 for 8-bit images)
- MSE: mean squared error between the original and reconstructed image

During the experiences, we find that there are some compressed images that are exactly the same as the original images, the score of PSNR will be infinite. To avoid getting an infinite average value, we set the maximum value of PSNR to 100. In practice, when PSNR reaches 50, the similarity is already very high.

$$PSNR' = \max(PSNR, 100) \tag{8}$$

4) NCC: The normalized cross-correlation (NCC) measures similarity between two images by comparing their normalized correlation in pixel patterns. It can avoid distribution from luminance and contrasts [2].

$$NCC = \frac{\sum(I(i,j) - \mu_I)(K(i,j) - \mu_K)}{\sqrt{\sum(I(i,j) - \mu_I)^2} \cdot \sqrt{\sum(K(i,j) - \mu_K)^2}} \tag{9}$$

where:

- $I(i,j)$: pixel value of the first image
- $K(i,j)$: pixel value of the second image
- μ_I, μ_K: mean intensity of images I and K

5) UIQI: The Universal Image Quality Index(UIQI) evaluates the similarity of images. It mainly focuses on assessing uniform brightness, contrast, and structural similarity [4].

$$UIQI(x,y) = \frac{4\sigma_{xy}\mu_x\mu_y}{(\sigma_x^2 + \sigma_y^2)(\mu_x^2 + \mu_y^2)} \tag{10}$$

where:

- μ_x, μ_y: mean intensity of images x and y
- σ_x^2, σ_y^2: variance of images x and y
- σ_{xy}: covariance between images x and y

6) DSSIM: The structural dissimilarity index (DSSIM) is derived from SSIM by converting similarity into dissimilarity. It measures the difference between two images instead of their similarity [1].

$$DSSIM(x, y) = \frac{1 - SSIM(x, y)}{2} \tag{11}$$

where:

- $SSIM(x, y)$: structural similarity index between images x and y

7) ResNet-18 and ResNet-50: Convolutional neural network with residual blocks. ResNet-18 uses the BasicBlock as its residual block and has approximately 11M parameters. ResNet-50 uses the Bottleneck structure for its residual blocks, has around 25M parameters, and typically achieves higher accuracy than ResNet-18. [5].

8) MobileNet-v2: A lightweight convolutional neural network, tailored for mobile and embedded devices, reduces the number of parameters while preserving high accuracy. It incorporates inverted residual blocks and linear bottlenecks, and has roughly 3.4M parameters [6].

C. Datasets

For datasets, we select CIFAR-100 and MINI-ImageNet, they both contain 100 different classes, we split them into train, test and validation datasets (Table I). We train with train datasets and evaluate our compression methods using test datasets.

TABLE I: Dataset Splits and Image Sizes

Dataset	Train	Test	Validation	Image Size
CIFAR-100 [7]	45000	10000	5000	32×32
Mini-ImageNet [8]	40000	5000	5000	$\sim 500 \times 400$

IV. Experiment

We evaluated compression methods using mathematical metrics and conventional networks. The Identity method means we directly return the original image when evaluating. In the case of PNG, no information is lost; thus, both methods can be considered as baselines. Since different methods have completely different ways to compress, we use a similar compressed size as a standard. For math calculation, we resize images to same pixel sizes.

A. Math Metrics

We tried different mathematical metrics with different datasets (from Figure 1 to Figure 6). With different metrics and datasets, the ranking of each method may vary. For JPEG, since it is a lossy compression method, it loses information even if we set the quality to 100%. For PNG and the Identity method, no information is lost; thus, they can be considered as baselines.

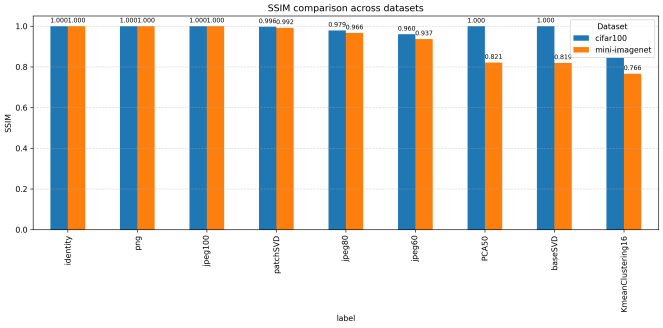

Fig. 1: SSIM score

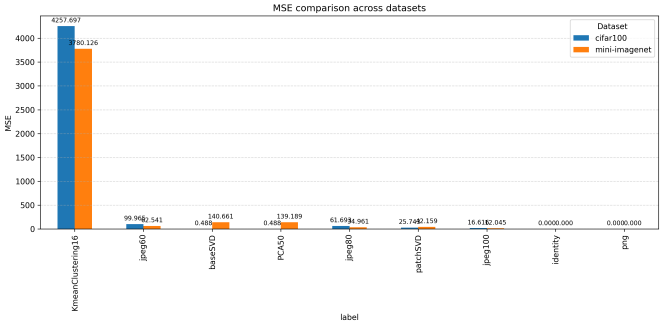

Fig. 2: MSE score

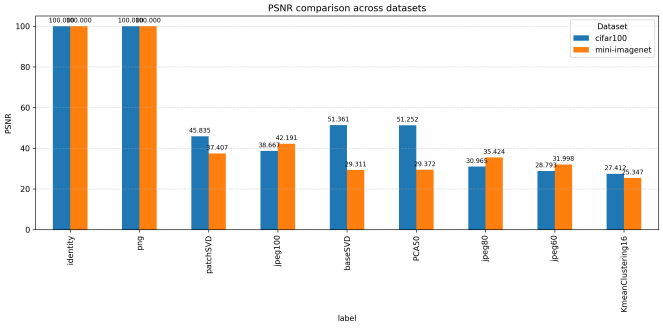

Fig. 3: PSNR score

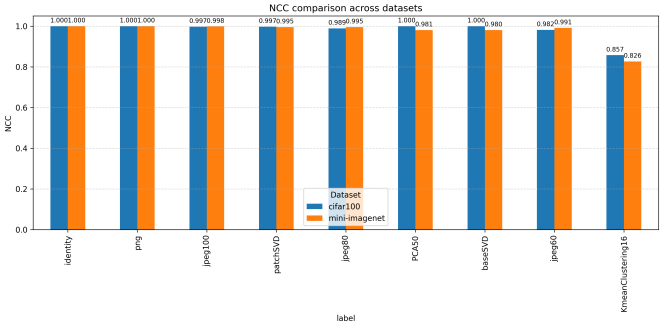

Fig. 4: NCC score

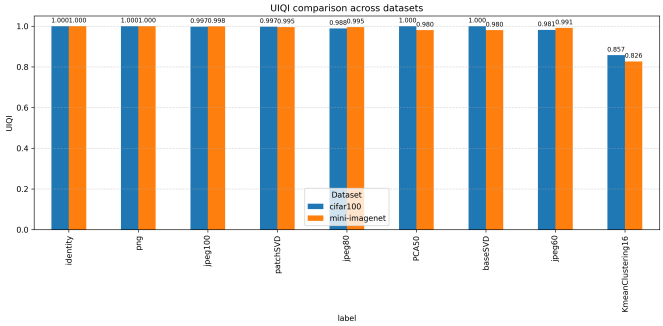

Fig. 5: UIQI score

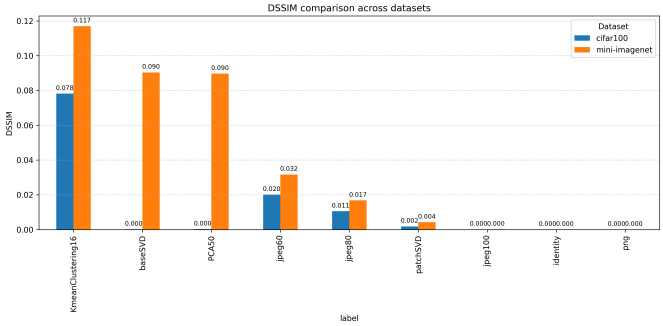

Fig. 6: DSSIM score

B. Computer Vision Models

1) Model Accuracy: Since the focus of this study is on evaluating compression methods, we did not place significant emphasis on model accuracy. Table II shows model accuracy with different datasets.

TABLE II: Validation and Test Accuracies of Models on CIFAR-100 and Mini-ImageNet

Dataset	Model	Validation Accuracy	Test Accuracy
CIFAR-100	ResNet-18	0.723	0.686
	ResNet-50	0.716	0.677
	MobileNetV2	0.705	0.674
Mini-ImageNet	ResNet-18	0.628	0.554
	ResNet-50	0.681	0.614
	MobileNetV2	0.609	0.541

2) Compressed Accuracy: We ran experiments on CIFAR-100 and Mini-ImageNet with different models, compression methods, and settings.

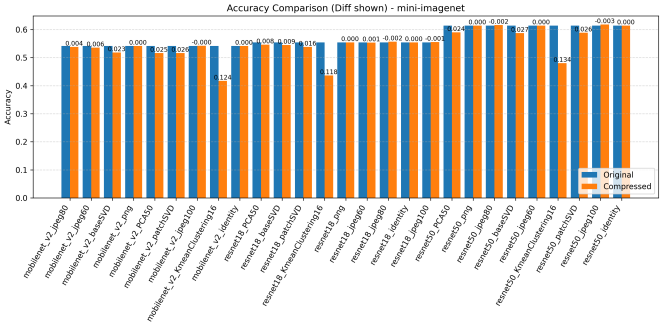

Fig. 7: compressed accuracy with MINI-ImageNet

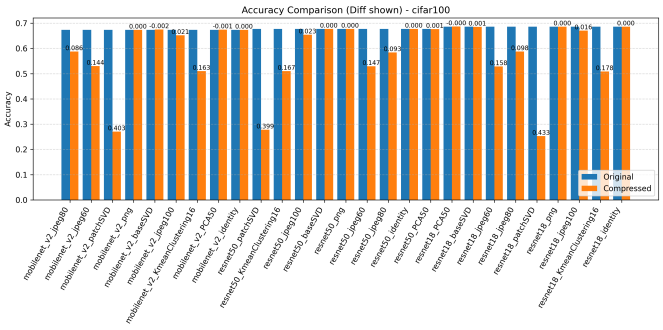

Fig. 8: compressed accuracy with CIFAR-100

Blue bars represent the original image accuracy, while orange bars represent the compressed accuracy (Figure 7 and Figure 8). The numbers at the top of the bars indicate the accuracy difference between before and after compression. Negative values mean that, after compression, the model accuracy improved. This phenomenon occurs across different compression methods, datasets, and models.

3) Further Experiment: From the previous section, we found that some compression methods may actually improve model accuracy; therefore, we conducted an experiment using the JPEG compression method.

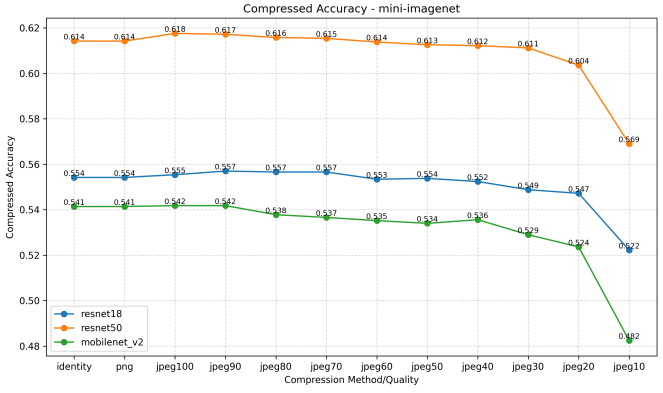

Fig. 9: Model Accuracy of MINI-ImageNet with JPEG Compression

Figure 9 shows the accuracy of ResNet-18, ResNet-50 and MobileNet-v2 with JPEG compression. Identity and PNG can be considered as baselines. This experiment shows that for all the models we have, with some compression, the image accuracy first improves, but after further compression, the accuracy drops.

C. Compression Time

We compare the compression time for each method. Since JPEG and PNG already have built-in packages in Python, they run faster than all the other methods.
Figure 10 and Figure 11 show the average compression time for images from the two datasets. During the experiments, we found that the compression time of some methods is not linear; it varies depending on image size and complexity, and the difference can be significant.

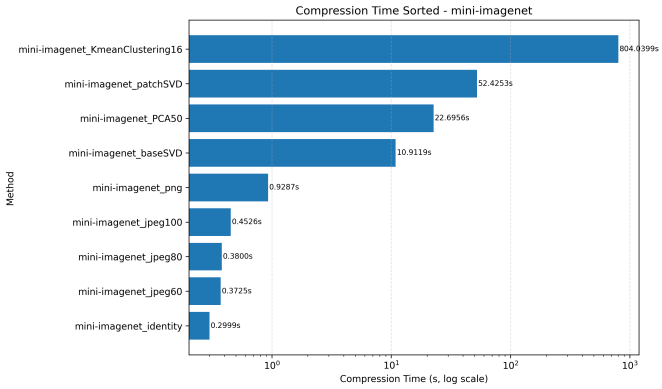

Fig. 10: Average Compression Time for MINI-ImageNet

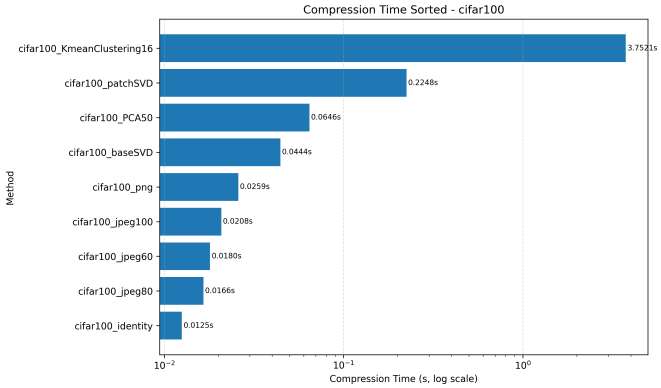

Fig. 11: Average Compression Time for CIFAR-100

D. Real Image Application(SVD and PatchSVD)

In order to find out the improvement of PatchSVD compared to normal SVD, we use the same complex real picture and set the same compression ratio for both algorithms and compare the results. When operating on the same image, the PatchSVD can subtract most of the features from the original image, while the SVD algorithm loses some information. The most significant part is that the text in the PatchSVD compressed image is still clear and readable, but in the SVD compressed image, it is hard to recognize.

(a) SVD (b) PatchSVD

Fig. 12: Compression result of SVD and PatchSVD

V. Conclusion

We compared PatchSVD with the SVD algorithm. The PatchSVD uses different k values for different patches, thus maintaining better compression results, and is able to handle high compression ratio tasks. However, the multiple SVD computations involved in PatchSVD make it a time-consuming algorithm. The significantly increased time complexity makes it unsuitable for very large images. We compared different compression methods with different metrics and datasets. The best compression method we found is JPEG compression at quality 100. It has the best score for most metrics and the shortest run time. PatchSVD is the next best, even outperforming JPEG at quality 80 and all other methods, though its run time is the second longest. KMeans Clustering is the worst method we found. It has the lowest score for most metrics and the longest run time. PCA and BaseSVD have similar performances; they are generally second tier.

A. Limitation

Due to time and computational resource limits, we could not run more experiments. Also, different methods have completely different compression algorithms, so it is hard to evaluate the compression size using a simple value. We tried our best to set a standard.

B. Future Work

We can run more experiments in the future by setting different standards for compression. In addition, the reason for accuracy improvement after compressing the image also needs to be discussed.

References

[1] Z. Wang, A. Bovik, H. Sheikh, and E. Simoncelli, "Image quality assessment: from error visibility to structural similarity," *IEEE Transactions on Image Processing*, vol. 13, no. 4, pp. 600–612, 2004.

[2] R. C. Gonzalez and R. E. Woods, *Digital Image Processing*, 3rd ed. Prentice Hall, 2008.

[3] A. Horé and D. Ziou, "Image quality metrics: Psnr vs. ssim," in *2010 20th International Conference on Pattern Recognition*, 2010, pp. 2366–2369.

[4] Z. Wang and A. Bovik, "A universal image quality index," *IEEE Signal Processing Letters*, vol. 9, no. 3, pp. 81–84, 2002.

[5] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 770–778.

[6] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "Mobilenetv2: Inverted residuals and linear bottlenecks," in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2018, pp. 4510–4520.

[7] A. Krizhevsky, "Learning multiple layers of features from tiny images," University of Toronto, Tech. Rep., 2009.

[8] O. Vinyals, C. Blundell, T. Lillicrap, K. Kavukcuoglu, and D. Wierstra, "Matching networks for one shot learning," in *Advances in Neural Information Processing Systems*, vol. 29, 2016.