AI in Image Processing

A Project Report Submitted
in Partial Fulfillment of the Requirements
for the Degree of

Bachelor of Technology

by

Kaushal Kishore
(111601008)

under the guidance of

Dr. Chandra Shekar

INDIAN INSTITUTE
OF TECHNOLOGY
PALAKKAD

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CERTIFICATE

*This is to certify that the work contained in this thesis entitled "**AI in Image Processing**" is a bonafide work of **Kaushal Kishore** (**Roll No. 111601008**), carried out in the Department of Computer Science and Engineering, Indian Institute of Technology Palakkad under my supervision and that it has not been submitted elsewhere for a degree.*

Dr. Chandra Shekar

Assistant Professor

Department of Computer Science & Engineering

Indian Institute of Technology Palakkad

Acknowledgements

Apart from the efforts of myself, the success of any project depends largely on the encouragement and guidelines of many others. I take this opportunity to express my gratitude to the people who have been instrumental in the successful completion of this project. I would like to show my greatest appreciation to Dr. Chandra Shekar. I can't say thank you enough for his tremendous support and help. I feel motivated and encouraged every time I attend his meeting. Without his encouragement and guidance, this project would not have materialized.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Image processing libraries these days (eg. Open CV) uses the conventional methods which have the possibility to be outperformed by methods which leverage the power of artificial intelligence. Some recent research have shown that some of these AI based methods are able to perform atleast as good as conventional approaches. The aim of this project is to implement, apply and possibly improve upon the existing approaches in Digital Image Processing and Computer Vision. These common tasks can include (not limited to) applications like: Image Compression, Denoising, Super Resolution, Flow Estimation, Object Detection, etc.

1.1 Image Processing

Image processing is manipulating an image in order to enhance it or extract information from it. It is widely useed in medical visualization, biometrics, self-driving vehicles, gaming, surveillance, and law enforcement. It can used in various ways: visualization, restoration, imformation retrieval, pattern recognition, etc.

General approach of image processing involves eight key phases: image acquisition, image enhancement, image restoration, color space transformation, compression or decom-

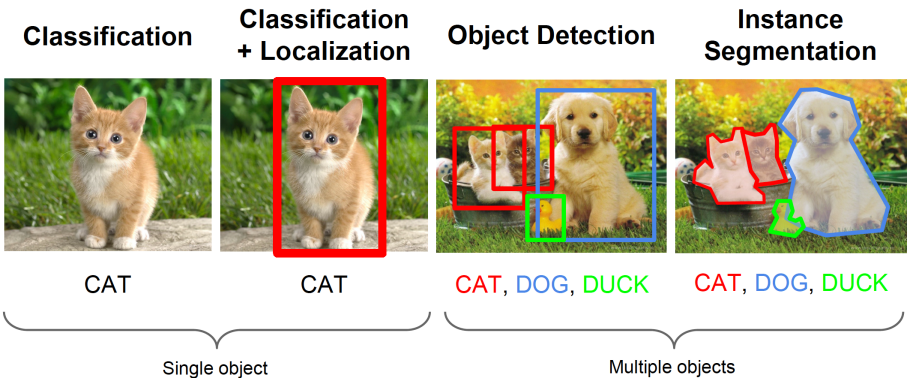

Fig. 1.1: Examples of pattern recognition
Source: www.cs.cornell.edu

pression, morphological processing, recognition, and representation. It is very difficult to carry out these steps manually on a very big data, this is where AI and ML algorithms become very helpful.

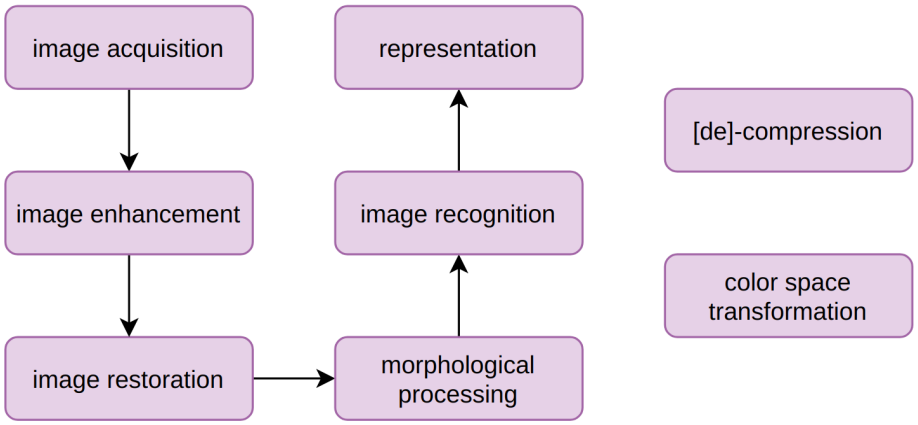

Fig. 1.2: Key phases of image processing

1.2 Recent advances

Modern AI algorithms have enabled computer to perform detection, segmentation, recognition, compression, extraction, generation, and discrimination. Every year a state of the art model is invented to solve the existing problem in a better way. It is now an established fact that machines are now better than humans in counting, classifying and segmenting instances.

1.3 Organization of The Report

Chapter 2

Image Compression

A data compression algorithm transforms the data to occupy a less space. The original data is encoded by a program called encoder, to a compressed representation using a fewer number of bits. Decoder is responsible for decompressing the compressed representation.

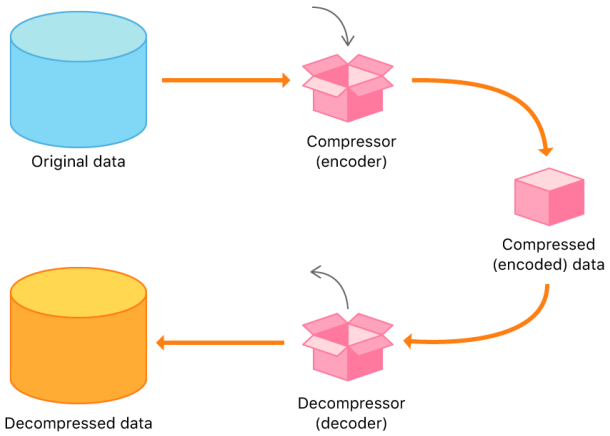

Fig. 2.1: Compression phases
Source: `https://developer.apple.com/documentation/compression`

Image compression is very crucial in order to reduce the size of disk space used as well as reduce the amount of internet bandwidth used while loading images. It's also important to compress images for people accessing the internet via low bandwidth connections.

2.1 Lossy vs. Lossless

The compression technique where the decompressed data is exactly same as original data is called as lossless compression otherwise it is known as lossy compression technique because some information is lost during coding-encoding phase. Two well-known codecs for image compression are JPEG and PNG. PNG is lossless and JPEG is lossy.

Lossy	Lossless
A compression that permits reconstruction only of an approximation of the original data, though usually with an improved compression rate.	A class of data compression that allows the original data to be perfectly reconstructed from the compressed data.
Reduces the quality.	Does not reduce the quality.
Data reduction is higher.	Data reduction is lower.
Commonly used to compress multimedia data such as audio (MP3), video and image (JPEG) files.	Used commonly for text, data files, etc.

Table 2.1: Lossy vs. Lossless

2.2 Dimensionality Reduction PCA

Principal components analysis (PCA) is one of a family of techniques for taking high-dimensional data, and using the dependencies between the variables to represent it in a more tractable, lower-dimensional form, without losing too much information. PCA is one of the simplest and most robust ways of doing such dimensionality reduction.

PCA is mathematically defined as an orthogonal linear transformation that transforms the data to a new coordinate system such that the greatest variance by some scalar projection of the data comes to lie on the first coordinate (called the first principal component),

the second greatest variance on the second coordinate, and so on.

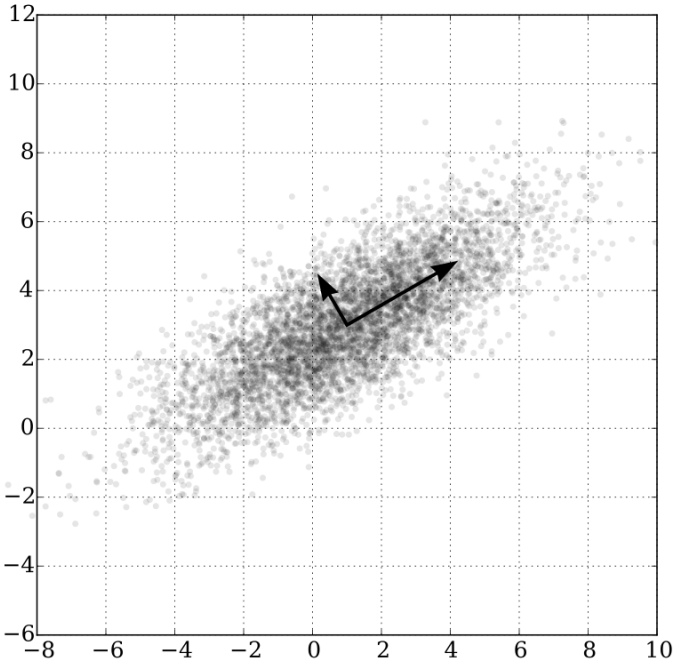

Fig. 2.2: PCA of a multivariate Gaussian distribution centered at (1,3) with a standard deviation of 3 in roughly the (0.866, 0.5) direction and of 1 in the orthogonal direction.
Source: `https://en.wikipedia.org/wiki/Principal_component_analysis`

Let W be a $d \times d$ matrix whose columns are the principal components of X. The transformation $T = XW$ maps a data vector $x_{(i)}$ from an original space of d variables to a new space of d variables which are uncorrelated over the dataset. However, not all the principal components need to be kept. Keeping only the first L principal components, produced by using only the first L eigenvectors, gives the truncated transformation:

$T_L = XW_L$

where the matrix T_L now has n rows but only L columns. In other words, PCA learns a linear transformation $t = W^T x, x \in \Re^d, t \in \Re^L$, where the columns of $d \times L$ matrix W form an orthogonal basis for the L features (the components of representation t) that are decorrelated. By construction, of all the transformed data matrices with only L columns,

this score matrix maximises the variance in the original data that has been preserved, while minimising the total squared reconstruction error $\left\|TW^T - T_L W_L^T\right\|_2^2 or \|X - X_L\|_2^2$.

The basic steps for computing the PCA is as follows:

1. Standardize the d-dimensional dataset.
2. Construct the covariance matrix.
3. Decompose the covariance matrix into its eigenvectors and eigenvalues.
4. Sort the eigenvalues by decreasing order to rank the corresponding eigenvectors.
5. Select L eigenvectors which correspond to the L largest eigenvalues, where L is the dimensionality of the new feature subspace $L \leq d$.
6. Construct a projection matrix W_L from the "top" L eigenvectors.
7. Transform the d-dimensional input dataset X using the projection matrix W_L to obtain the new L-dimensional feature subspace.

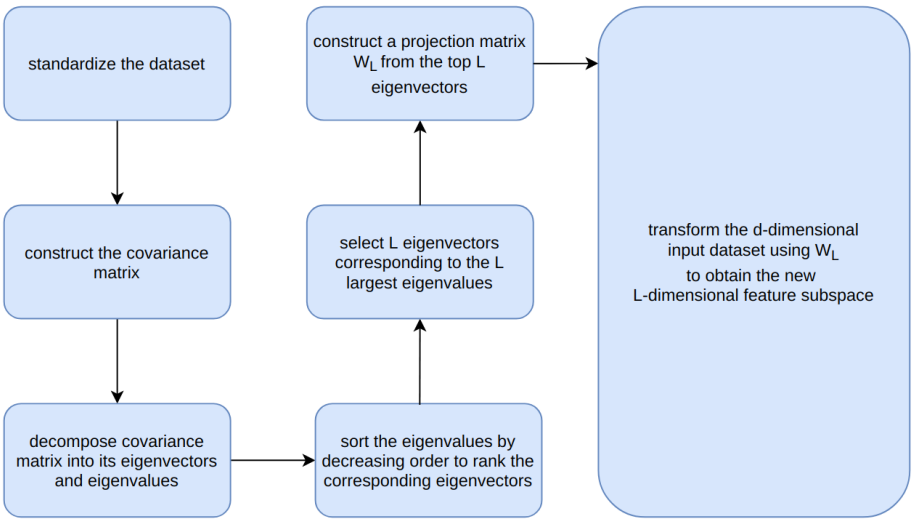

Fig. 2.3: Steps for dimensionality reduction using PCA

2.3 Image compression using PCA

We will follow the steps mentioned in the previous section for dimensionality reduction using PCA. The results of the experiments are tabulated below:

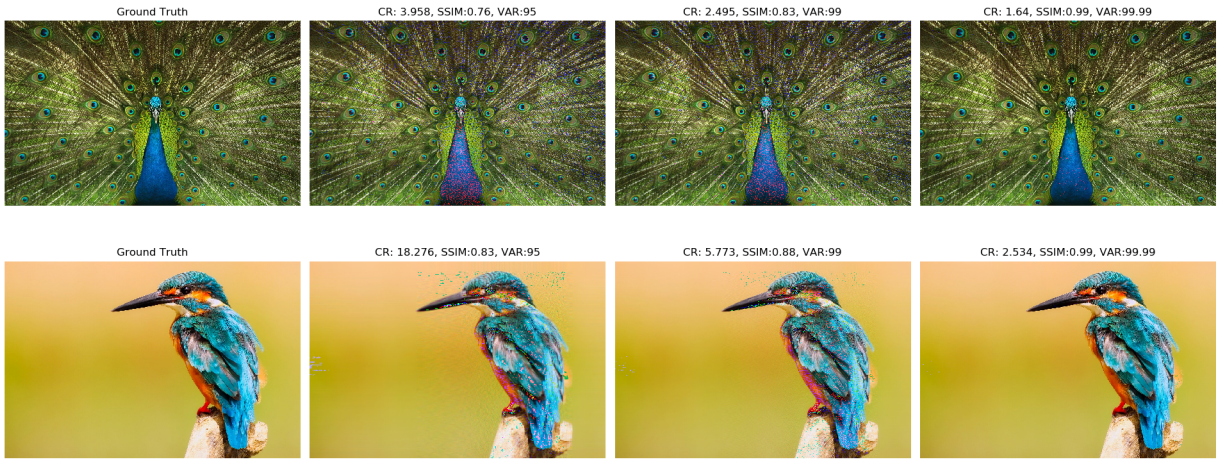

Fig. 2.4: CR = Compression Ratio, SSIM = Structural Similarity Index, VAR = Variance. Results of compression using the PCA method. Higher variance leads to more number of principal components and higher is the reconstructed image quality and lower the compression rate.

Code for the above test can be found here: @KishoreKaushal/ImageCompression/

It is clear from the above experiment that higher variance leads to more number of principal components and higher is the reconstructed image quality and lower the compression rate. Infact, the first L principal components is selected to get atleast the given number of variance.

In the next chapter we will discuss the JPEG compression algorithm.

Chapter 3

JPEG

3.1 Overview

JPEG, which stands for Joint Photographic Experts Group (the name of the committee that created the JPEG standard) is a lossy compression algorithm for images. A lossy compression scheme is a way to inexactly represent the data in the image, such that less memory is used yet the data appears to be very similar. This is why JPEG images look almost the same as the original images they were derived from most of the time unless the quality is reduced significantly, in which case there will be visible differences.

The JPEG algorithm takes advantage of the fact that humans can't see colors at high frequencies. These high frequencies are the data points in the image that are eliminated during the compression. JPEG compression also works best on images with smooth colour transitions.

JPEG is a commonly used method of lossy compression for digital images. The degree of compression can be adjusted, allowing a the tradeoff between storage size and image quality with a compression ratio 10:1 but with little perceptible loss in image quality.

Fig. 3.1: A photo of a European wildcat with the compression rate decreasing and hence quality increasing, from left to right.
Source: `https://en.wikipedia.org/wiki/JPEG`

3.2 Algorithm

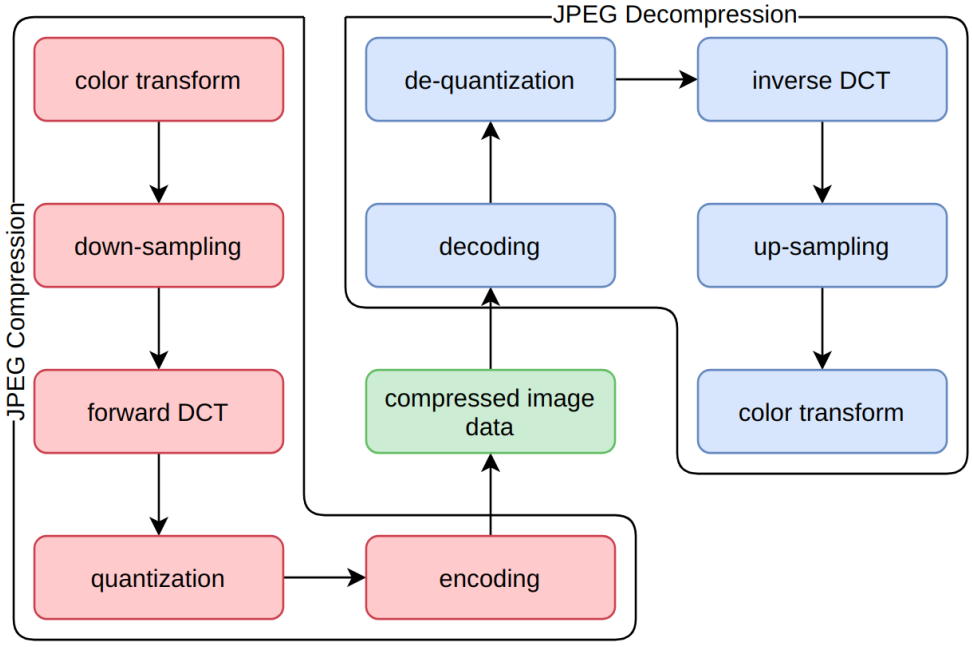

Fig. 3.2: JPEG Schematic

3.2.1 Splitting

JPEG uses transform coding, it is largely based on the following observations:

- Usually image contents change relatively slowly across images, i.e., it is unusual for intensity values to alter up and down several times in a small area, for example, within an 8×8 image block. A translation of this fact into the spatial frequency domain, implies, generally, lower spatial frequency components contain more information than the high frequency components which often correspond to less useful details and noises.

- Experiments suggest that humans are more immune to loss of higher spatial frequency components than loss of lower frequency components. Human vision is insensitive to high frequency components.

Hence, first step is to split the image into 8×8 blocks non-overlapping pixel blocks. If the image cannot be divided into 8×8 blocks then the block is zero padded.

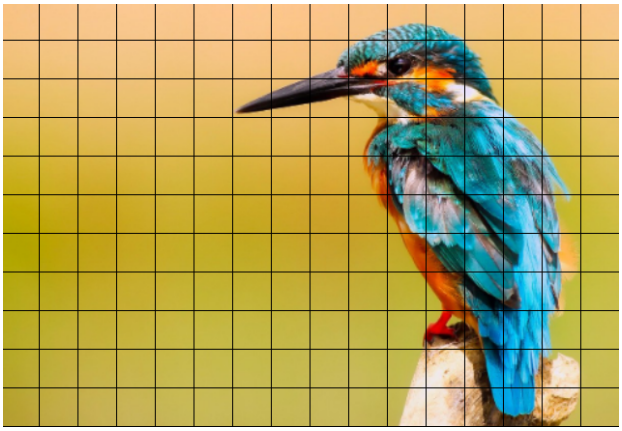

Fig. 3.3: Splitting into 8×8 blocks

3.2.2 Color Space Transform

JPEG makes use of $[Y, Cb, Cr]$ model instead of $[R, G, B]$ model. There is a reason for using the color space. The human eye is more sensitive to luminance than to chrominance.

$Y = 0.299R + 0.587G + 0.114B$
$Cb = -0.1687R - 0.3313G + 0.5B + 128$
$Cr = 0.5R - 0.4187G - 0.0813B + 128$
$R = Y + 1.402(Cr - 128)$
$G = Y - 0.34414(Cb - 128) - 0.71414(Cr - 128)$
$B = Y + 1.772(Cb - 128)$

Table 3.1: RGB and YCbCr conversion table

3.2.3 Down-sampling

After the color space transformation, the image is downsampled. Downsampling happens in such a way that Y is taken for each pixel whereas Cb and Cr are taken for 2×2 blocks.

Sometimes this step is not carried out(baseline JPEG).

3.2.4 Discrete Cosine Transform

The key to the JPEG baseline compression process is a mathematical transformation known as the Discrete Cosine Transform (DCT). The DCT is in a class of mathematical operations that includes the well known Fast Fourier Transform (FFT), as well as many others. The basic purpose of these operations is to take a signal and transform it from one type of representation to another. For example, an image is a two-dimensional signal that is perceived by the human visual system. The DCT can be used to convert the signal (spatial information) into numeric data ("frequency" or "spectral" information) so that the image's information exists in a quantitative form that can be manipulated for compression.

The DCT is performed on a $N \times N$ square matrix of pixel values, and it yields a $N \times N$ square matrix of frequency coefficients. (In practice, N most often equals 8 because a larger block, though would probably give better compression, often takes a great deal of time to perform DCT calculations, creating an unreasonable tradeoff. As a result, DCT implementations typically break the image down into more manageable 8×8 blocks.) The

DCT formula 3.1 looks somewhat intimidating at first glance but can be implemented with a relatively straightforward piece of code.

$$DCT[i,j] = \frac{1}{\sqrt{2N}}\, C[i]\, C[j] \sum_{x=0}^{N-1} \sum_{y=0}^{N-1} I[x,y]\, cos[\frac{(2x+1)i\pi}{2N}]\, cos[\frac{(2y+1)j\pi}{2N}] \\ where\ C(x) = \frac{1}{\sqrt{2}}\ if\ x = 0,\ else\ 1\ if\ x > 0 \tag{3.1}$$

Below are two matrices representing the DCT input 3.2 and DCT output blocks 3.3 from a gray-scale image. Each element of the 8-pixel-by-8-pixel input matrix contains the value of the pixel at the corresponding (x, y) location. These integer values are fed to the DCT algorithm, creating the output matrix shown below it. Each element of the output matrix is a coefficient by which the waveform of the corresponding spatial frequency is multiplied in the decomposition of the image sample.

$$\begin{bmatrix} 140 & 144 & 147 & 140 & 140 & 155 & 179 & 175 \\ 144 & 152 & 140 & 147 & 140 & 148 & 167 & 179 \\ 152 & 155 & 136 & 167 & 163 & 162 & 152 & 172 \\ 168 & 145 & 156 & 160 & 152 & 155 & 136 & 160 \\ 162 & 148 & 156 & 148 & 140 & 136 & 147 & 162 \\ 147 & 167 & 140 & 155 & 155 & 140 & 136 & 162 \\ 136 & 156 & 123 & 167 & 162 & 144 & 140 & 147 \\ 148 & 155 & 136 & 155 & 152 & 147 & 147 & 136 \end{bmatrix} \tag{3.2}$$

Why DCT? Human vision is insensitive to high-frequency components, due to which it is possible to treat the data corresponding to high frequencies as redundant. To segregate the raw image data on the basis of frequency, it needs to be converted into frequency domain, which is the primary function of DCT.

$$\begin{bmatrix} 1209 & -17 & 14 & -8 & 23 & -9 & -13 & -18 \\ 20 & -34 & 26 & -9 & -10 & 10 & 13 & 6 \\ -10 & -23 & -1 & 6 & -18 & 3 & -20 & 0 \\ -8 & -5 & 14 & -14 & -8 & -2 & -3 & 8 \\ -3 & 9 & 7 & 1 & -10 & 17 & 18 & 15 \\ 3 & -2 & -18 & 8 & 8 & -3 & 0 & -6 \\ 8 & 0 & -2 & 3 & -1 & -7 & -1 & -1 \\ 0 & -7 & -2 & 1 & 1 & 4 & -6 & 0 \end{bmatrix} \tag{3.3}$$

3.2.5 Coefficient Quantization

Humans are unable to see aspects of an image that are at really high frequencies. Since taking the DCT allows us to isolate where these high frequencies are, we can take advantage of this in choosing which values to preserve. By multiplying the DCT matrix by some mask, we can zero out elements of the matrix, thereby freeing the memory that had been representing those values. The resultant quantize matrix will only preserve values at the lowest frequencies up to a certain point.

$$DCT_q[i,j] = \left\lfloor \frac{DCT[i,j]}{Q[i,j]} \right\rfloor \tag{3.4}$$

In Eq. 3.4 Q is the quantization matrix. JPEG Standard defines two default quantization tables, one each for luminance and chrominance. From the formula, one can notice the smaller DCT coefficients of high-frequency elements divided by the larger quantum values will most often result in the high-frequency coefficients being rounded down to zero.

16	11	10	16	24	40	51	61
12	12	14	19	26	58	60	55
14	13	16	24	40	57	69	56
14	17	22	29	51	87	80	62
18	22	37	56	68	109	103	77
24	35	55	64	81	104	113	92
49	64	78	87	103	121	120	101
72	92	95	98	112	100	103	99

Table 3.2: Luminance Quantization Table

17	18	24	47	99	99	99	99
18	21	26	66	99	99	99	99
24	26	56	99	99	99	99	99
47	66	99	99	99	99	99	99
99	99	99	99	99	99	99	99
99	99	99	99	99	99	99	99
99	99	99	99	99	99	99	99
99	99	99	99	99	99	99	99

Table 3.3: Chrominance Quantization Table

The quantization step is the major information losing step in JPEG compression.

3.2.6 Encoding

The final step of the JPEG image compression process is to compress the quantized DCT values. This is done through a three-part procedure detailed below.

First, the DC coefficient is changed from absolute value to a relative value – relative to the DC coefficient in the previous 8×8 block, that is. Since adjacent blocks in an image

exhibit a high degree of correlation, coding the DC term of a given block as the difference from the previous DC term typically produces a very small number that can be stored in a fewer number of bits.

Second, reorder the DCT block in a zig-zag sequence. Because so many coefficients in the DCT image are truncated to zero values during the coefficient quantization stage, the zeros are handled differently than non-zero coefficients. They are coded using a Run-Length Encoding (RLE) algorithm. RLE gives a count of consecutive zero values in the image, and the longer the runs of zeros, the greater the compression. One way to increase the length of runs is to reorder the coefficients in the zig-zag sequence shown in the diagram below. This way, the JPEG algorithm moves through the block selecting the highest value elements first and eventually working its way to the lowest value elements, thus optimizing the effect of RLE.

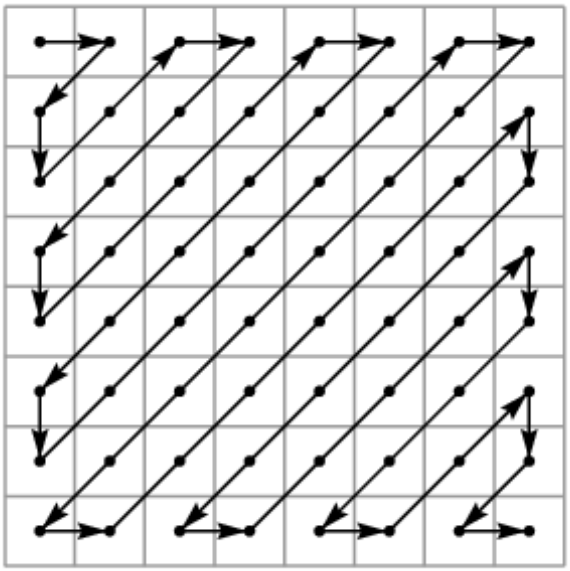

Fig. 3.4: Reorder the DCT block in a zig-zag sequence
Source: freecodecamp.org/

Third, Entropy Encoding. Finally, the JPEG algorithm outputs the DCT block's elements using an entropy encoding mechanism that combines the principles of RLE and Huffman encoding. The output of the entropy encoder consists of a sequence of three

tokens, repeated until the block is complete. The three tokens are the run length, the number of consecutive zeros that precede the current non-zero element in the DCT output matrix; the bit count, the number of bits used to encode the amplitude value that follows, as determined by the Huffman encoding scheme; and the amplitude, the amplitude of the DCT coefficient.

The final output that results from JPEG compression can be mathematically reversed and decompressed to produce an image that, to a computer, lacks defining data, but to a human eye, it appears identical to the original image.

3.3 Conclusion

In this chapter, we discussed a well-known image compression algorithm. In the next chapter, we will introduce a deep learning model which modifies the existing JPEG compression a bit and makes compression better than JPEG.

Chapter 4

Perceptual Image Compression

Prakash2017 [1] introduced a powerful CNN tailored to the specific task of semantic image understanding to achieve higher visual quality in lossy compression. A modest increase in complexity is incorporated into the encoder which allows a standard, off-the-shelf jpeg decoder to be used. While JPEG encoding may be optimized for generic images, the process is ultimately unaware of the specific content of the image to be compressed. This technique makes JPEG content-aware by designing and training a model to identify multiple semantic regions in a given image.

4.1 Content aware compression

The existing compression standards are content unware. For example, consider the case of JPEG algorithm, during the quantization step the quantization tables that are used are experimentally decided on a well-established theory that humans are insensitive to chrominance in contrast to luminance. An obvious question is can't we have a quantization table tailored specifically for each image. Of course, determining this table manually is a very uncertain and difficult task, and its well suited for the machine to do this for us.

The idea here is to locate multiple regions of interest (ROI) within a single image and noting the fact that it's not an object detection problem and hence the precision of the

boundary doesn't matter. Also, the model needs to learn a single class-invariant feature map by learning separate feature maps for each of a set of object classes and then summing over the top features.

4.2 Object Localization using CNNs

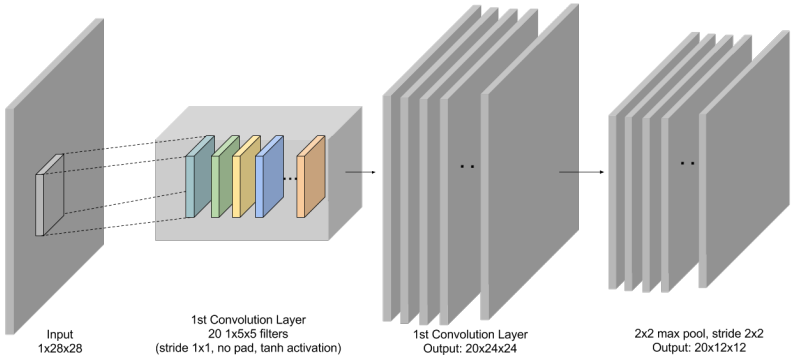

Fig. 4.1: First convolutional layer in LeNet
Source: https://mxnet.apache.org/

CNNs are multi-layered feed-forward architectures where the learned features at each level are the weights of the convolution filters to be applied to the output of the previous level. Learning is done via gradient-based optimization. CNNs differ from FCNNs in that the dimensions of the learned convolution filters are, in general, much smaller than the dimensions of the input image, so the learned features are forced to be localized in space. Also, the convolution operation uses the same weight kernel at every image location, so feature detection is spatially invariant.

CNNs include a max-pooling step after every or every other layer of convolution, in which the height and width of the feature map (filter response) are reduced by replacing several neighboring activations (coefficients), generally within a square window, with a single activation equal to the maximum within that window.

This pooling operation is strided, but the size of the pooling window can be greater than the stride, so windows can overlap. This results in down-sampling of input data, and

filters applied to such a map will have a larger receptive field (spatial support in the pixel space) for a given kernel size, thus reducing the number of parameters of the CNN model and allowing the training of much deeper networks. This does not change the depth of the feature map, but only its width and height. In practice, pooling windows are generally of size 2×2 or 4×4, with a stride of two, which reduces the number activations by 75%.

Region-based CNNs use a moving window to maximize the posterior of the presence of an object. Recently many approaches have been proposed for object localization but either they are computationally expensive or they are limited to determine the presence/absence of a single class of object within an image. Moving-window methods can localize multiple objects but can't produce object silhouettes. In contrast, recent deep learning models proposed for semantic segmentation can produce a close object boundary but do not scale well for more than a small number of object categories. The proposed method only requires a rough idea but clear enough image-object map.

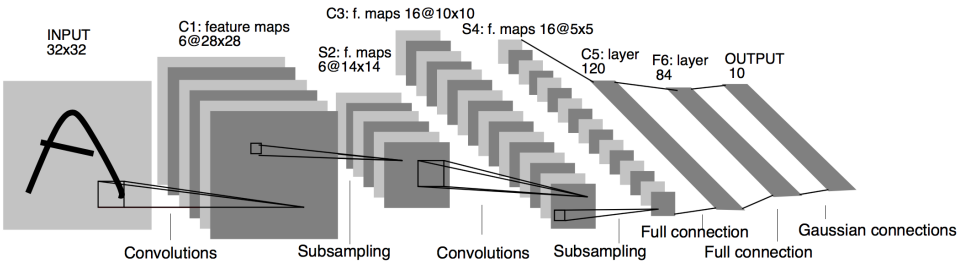

Fig. 4.2: LeNet : a traditional CNN
Source: medium.com/@pechyonkin/

Unlike traditional CNNs where the last two layers are fully-connected, the authors of [2] and [3] modified the second to last layer to allow for class localization.

4.3 Multi-Structure Region of Interest

Since CAMs are trained to maximize the posterior probability for the class, they tend to only highlight a single most prominent object. This makes them useful for studying the

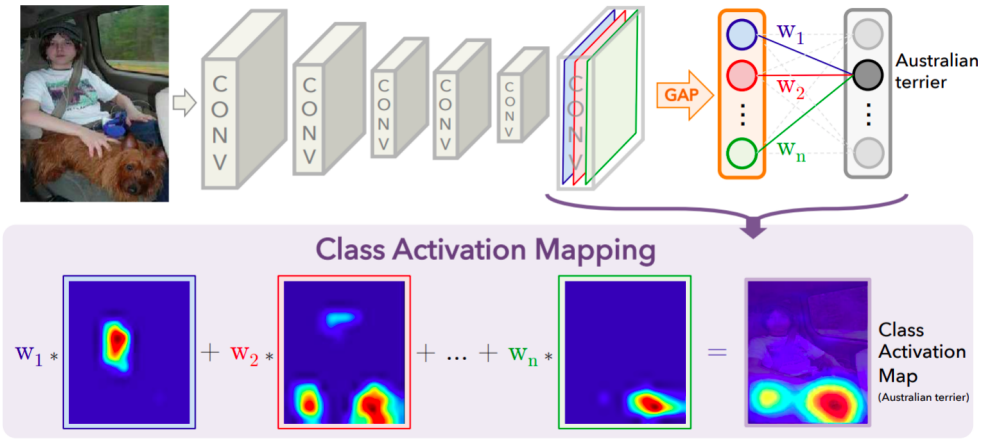

Fig. 4.3: Class Activation Mapping: the predicted class score is mapped back to the previous convolutional layer to generate the class activation maps (CAMs). The CAM highlights the class-specific discriminative regions.
Source: Zhou2016 [3]

output of CNNs, but not well suited to more general semantic saliency, as real-world images typically contain multiple objects of interest.

The authors of [1] have proposed a developed a variant of CAM which balances the activation for multiple objects and thus does not suffer from the issues of global average pooling. MS-ROI allows us to effectively train on localization tasks independent of the number of classes.

4.4 MS-ROI and JPEG

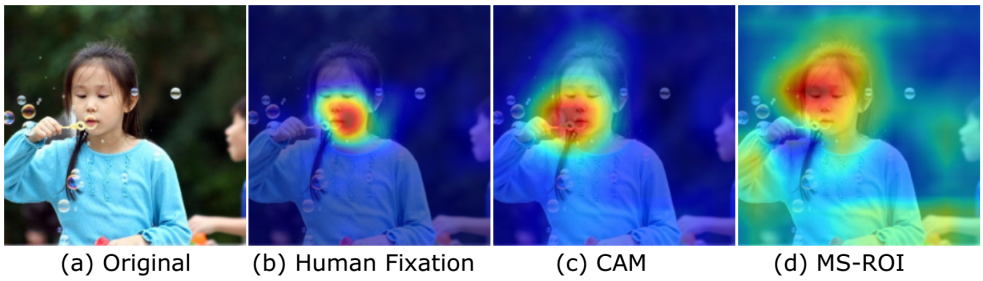

Fig. 4.4: Comparison of various methods of detecting objects in an image
Source: Prakash2017 [1]

MS-ROI returns a saliency value for each pixel in the range of $[0, 1]$, where 1 means

maximum saliency. Using this saliency values the quality factor is decided, which will finally be used to determine the quantization matrix.

4.5 Conclusion

In this chapter we have discussed a method which modifies the JPEG algorithm such that the final compression algorithm is content aware. The method can be integrated effortlessly with any off-the-shelf JPEG codec.

Chapter 5

Conclusion and Future Work

In this report, we first highlighted a conventional approach for image compression and then we discussed an AI based modification. In JPEG compression a suitable quality factor is set at the beginning with no knowledge of the compression ratio versus image quality. In contrast, the latter method provided a way by which a suitable quality factor is automatically decided by the model.

References

[1] A. Prakash, N. Moran, S. Garber, A. DiLillo, and J. Storer, "Semantic perceptual image compression using deep convolution networks," 2017.

[2] M. Oquab, L. Bottou, I. Laptev, and J. Sivic, "Is object localization for free? weakly-supervised learning with convolutional neural networks," *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 685–694, 2015.

[3] B. Zhou, A. Khosla, A. Lapedriza, A. Oliva, and A. Torralba, "Learning deep features for discriminative localization," *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.