Veri Depolama ve Sıkıştırma Algoritmaları

Bursa Teknik Üniversitesi
Bilgisayar Mühendisliğine Giriş Dersi
Muhammed Emin Karakurt
Bilgisayar Mühendisliği
Öğrenci no:25360859086

İÇERİK

- 1.Metin,Resim ve Ses Verilerinin Bit Düzeyinde Temsili
- 1.1 Metin Verilerinin Temsili
- 1.2 Resim Verilerinin Temsili
- 1.3 Ses Verilerinin Temsili

- 2. Veri Sıkıştırmanın Önemi
- 2.1 Depolama Alanında Tasarruf
- 2.2 İletim Hızı ve Bant Genişliği
- 2.3 Maliyetin Azaltılması
- 2.4 Donanım Performansı
-

3.Temel Kod Sıkıştıma Yöntemleri/Mantıkları

3.1 RLE Mantığı

3.2 Sözlük Tabanlı Sıkıştırma

3.3 Huffman Kodlaması

Blok Bazlı Sıkıştırma

1.Metin,Resim ve Ses Verilerinin Bit Düzeyinde Temsili:

- Dijital dünyada gördüğümüz(resim,fotoğraf...), duyduğumuz(müzik...) ve okuduğumuz(makale,dijital kitap) her şey, en temel seviyede **0** ve **1** rakamlarından oluşan **bit** dizileridir. Bilgisayarlar bu verileri işlemek için farklı kodlama yöntemleri kullanır.

1.1. Metin Verilerinin Temsili (Karakter Kodlama)

- Metinler, her bir harf veya sembolün sayısal bir değere atanmasıyla ikili sisteme dönüştürülür.

- **ASCII:** En eski standartlardan biridir. Her karakteri 7 bit (genelde 8 bite tamamlanır) ile ifade eder. Örneğin, 'A' harfi ASCII tablosunda **65** sayısına denk gelir ve bit karşılığı 01000001'dir.

?ASCII 7 bitten 8 bite?

1. Başa **"0"** ekleyerek: En yaygın yöntem budur. 7 bitlik ASCII kodunun soluna bir adet **"0"** eklenir.Bu karakterin değerini değiştirmez.
- Örnek: **7 Bitlik 'A' harfi:** 1000001 (Ondalık karşılığı 65)
- **8 Bitlik 'A' harfi:** 01000001 (Başa 0 eklendi)

2) Parite Biti(Hata Kontrolü)
3)Genişletilmiş ASCII

- **Unicode (UTF-8):** Günümüzün standardıdır. Sadece İngiliz harflerini değil, dünyadaki tüm dilleri, emojileri ve sembolleri kapsar. Değişken uzunlukta (8 ile 32 bit arası) veri kullanır.

- **Örnek:**
- M » U+004D(77) »01001101
- Bu Dönüşüm Nasıl Yapılır?
  1.Kod Noktasını Bul: Örneğin 'M' harfi için bu U+004D'dir.
  2.Onluk Sisteme Çevir: Hexadecimal (onaltılık) tabandaki 4D, onluk tabanda 77 sayısına eşittir.
  3.İkili Sisteme Çevir: 77 sayısı ikili sisteme çevrildiğinde 1001101 olur.

  4.8 Bite Tamamla: Başına bir adet 0 eklenerek 8 bitlik (1 bayt) standart yapıya ulaştırılır: 01001101

- Bu 1 ve 0 dizileri, bilgisayarınızın işlemcisinde düşük ve yüksek voltaj sinyalleri olarak işlenir ve ekranda anlamlı harfler olarak belirir.

ASCII TABLE

Decimal	Hexadecimal	Binary	Octal	Char	Decimal	Hexadecimal	Binary	Octal	Char	Decimal	Hexadecimal	Binary	Octal	Char
0	0	0	0	*[NULL]*	48	30	110000	60	0	96	60	1100000	140	`
1	1	1	1	*[START OF HEADING]*	49	31	110001	61	1	97	61	1100001	141	a
2	2	10	2	*[START OF TEXT]*	50	32	110010	62	2	98	62	1100010	142	b
3	3	11	3	*[END OF TEXT]*	51	33	110011	63	3	99	63	1100011	143	c
4	4	100	4	*[END OF TRANSMISSION]*	52	34	110100	64	4	100	64	1100100	144	d
5	5	101	5	*[ENQUIRY]*	53	35	110101	65	5	101	65	1100101	145	e
6	6	110	6	*[ACKNOWLEDGE]*	54	36	110110	66	6	102	66	1100110	146	f
7	7	111	7	*[BELL]*	55	37	110111	67	7	103	67	1100111	147	g
8	8	1000	10	*[BACKSPACE]*	56	38	111000	70	8	104	68	1101000	150	h
9	9	1001	11	*[HORIZONTAL TAB]*	57	39	111001	71	9	105	69	1101001	151	i
10	A	1010	12	*[LINE FEED]*	58	3A	111010	72	:	106	6A	1101010	152	j
11	B	1011	13	*[VERTICAL TAB]*	59	3B	111011	73	;	107	6B	1101011	153	k
12	C	1100	14	*[FORM FEED]*	60	3C	111100	74	<	108	6C	1101100	154	l
13	D	1101	15	*[CARRIAGE RETURN]*	61	3D	111101	75	=	109	6D	1101101	155	m
14	E	1110	16	*[SHIFT OUT]*	62	3E	111110	76	>	110	6E	1101110	156	n
15	F	1111	17	*[SHIFT IN]*	63	3F	111111	77	?	111	6F	1101111	157	o
16	10	10000	20	*[DATA LINK ESCAPE]*	64	40	1000000	100	@	112	70	1110000	160	p
17	11	10001	21	*[DEVICE CONTROL 1]*	65	41	1000001	101	A	113	71	1110001	161	q
18	12	10010	22	*[DEVICE CONTROL 2]*	66	42	1000010	102	B	114	72	1110010	162	r
19	13	10011	23	*[DEVICE CONTROL 3]*	67	43	1000011	103	C	115	73	1110011	163	s
20	14	10100	24	*[DEVICE CONTROL 4]*	68	44	1000100	104	D	116	74	1110100	164	t
21	15	10101	25	*[NEGATIVE ACKNOWLEDGE]*	69	45	1000101	105	E	117	75	1110101	165	u
22	16	10110	26	*[SYNCHRONOUS IDLE]*	70	46	1000110	106	F	118	76	1110110	166	v
23	17	10111	27	*[ENG OF TRANS. BLOCK]*	71	47	1000111	107	G	119	77	1110111	167	w
24	18	11000	30	*[CANCEL]*	72	48	1001000	110	H	120	78	1111000	170	x
25	19	11001	31	*[END OF MEDIUM]*	73	49	1001001	111	I	121	79	1111001	171	y
26	1A	11010	32	*[SUBSTITUTE]*	74	4A	1001010	112	J	122	7A	1111010	172	z
27	1B	11011	33	*[ESCAPE]*	75	4B	1001011	113	K	123	7B	1111011	173	{
28	1C	11100	34	*[FILE SEPARATOR]*	76	4C	1001100	114	L	124	7C	1111100	174	|
29	1D	11101	35	*[GROUP SEPARATOR]*	77	4D	1001101	115	M	125	7D	1111101	175	}
30	1E	11110	36	*[RECORD SEPARATOR]*	78	4E	1001110	116	N	126	7E	1111110	176	~
31	1F	11111	37	*[UNIT SEPARATOR]*	79	4F	1001111	117	O	127	7F	1111111	177	*[DEL]*

1.2.Resim Verilerinin Temsili(Piksel ve Renk)

Piksel Yapısı: Bir resim, satır ve sütunlardan oluşan bir matris gibidir.

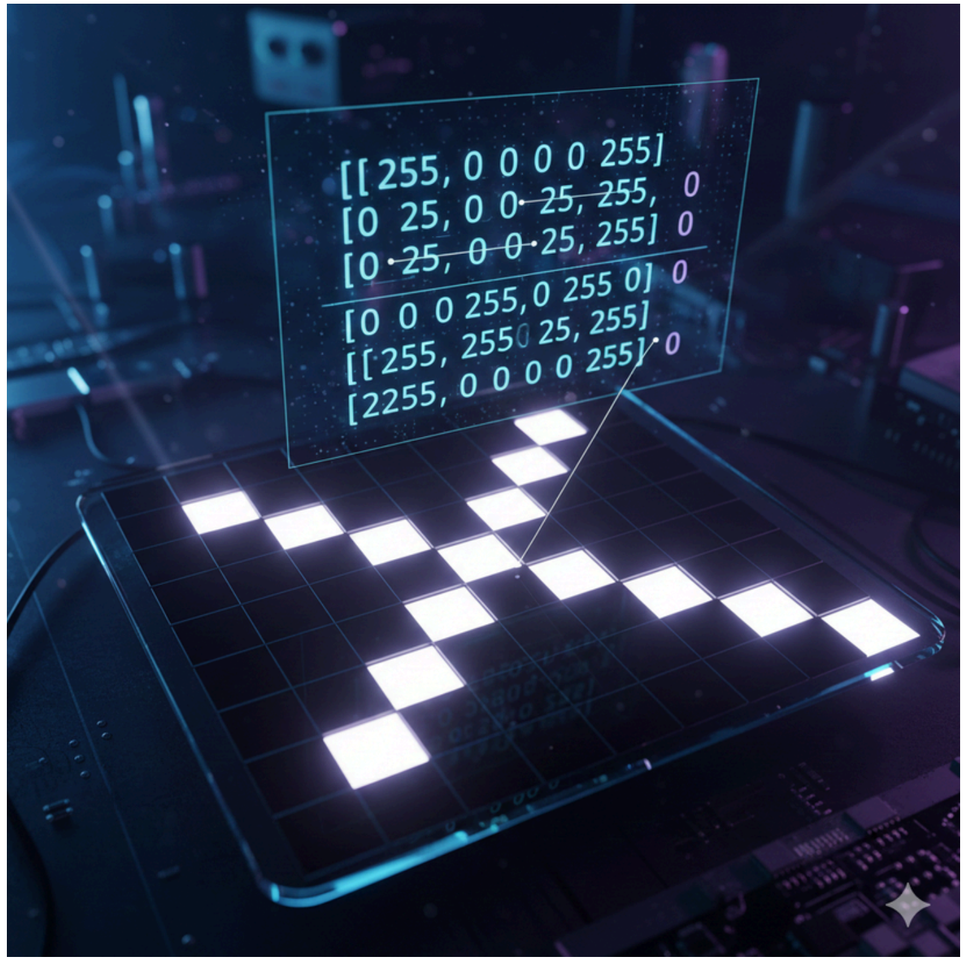
Renk Derinliği (RGB): En yaygın yöntem olan RGB (Red, Green, Blue) sisteminde her piksel üç ana rengin karışımından oluşur.

. Her renk kanalı genellikle 8 bit (1 bayt) ile temsil edilir.
.Bu durumda bir bit için 8x3=24 bit kullanılır.
.Örneğin saf kırmızı:11111111(R) -00000000(G)-00000000(B)olarak kodlanır.

X resmi kodu

- import numpy as np

- # 0 = Siyah, 255 = Beyaz
- resim_matrisi = np.array([
- [255, 0, 0, 0, 255],
- [0, 255, 0, 255, 0],
- [0, 0, 255, 0, 0],
- [0, 255, 0, 255, 0],
- [255, 0, 0, 0, 255]
-])



1.3. Ses Verilerinin Temsili (Örnekleme)

- Ses, doğası gereği analog bir dalgadır. Bunu dijital (bit) hale getirmek için **Örnekleme (Sampling)** ve **Kuantalama** süreçleri kullanılır.
- **Örnekleme Hızı (Sampling Rate):** Saniyede kaç kez ses dalgasının yüksekliğinin ölçüldüğünü ifade eder (Örn: CD kalitesi için saniyede 44.100 kez).
- Bit Derinliği:Her bir örneğin ne kadar hassas kaydedileceğidir.16 bitlik bir derinlikte,ses dalgasının yüksekliği 2^16=65.536 farklı seviyeden biriyle temsil edilir.
- Sonuçta, dalga üzerindeki her bir nokta, o anki genliği temsil eden bir ikili sayıya (bit dizisine) dönüşür.

Psikoakustik: Kulak Neyi Duymaz?

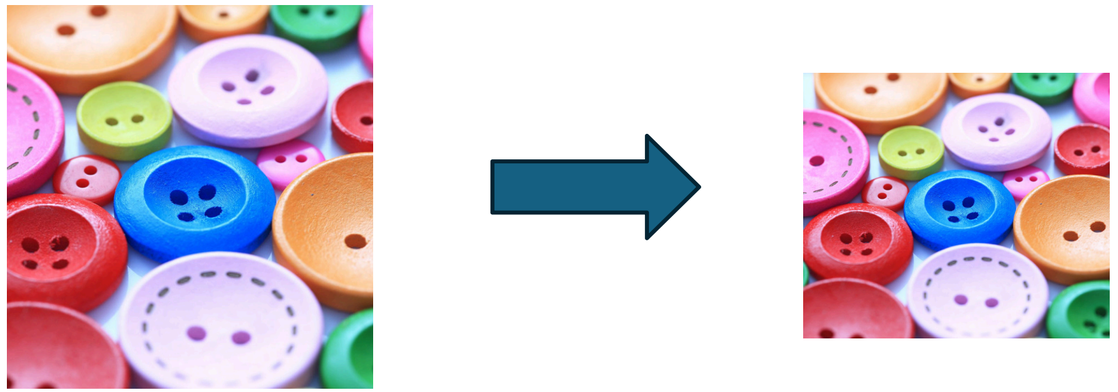
- Ham ses verisi (WAV) çok yer kaplar. MP3 gibi formatlar **Psikoakustik Modelleme** kullanarak veriyi küçültür. Mantık şudur: "Eğer beyin bu sesi duymuyorsa, veriyi saklamaya gerek yok.«
- **Eşik Altı Sesler:** İnsan kulağı 20 Hz ile 20 kHz arasını duyar. Bu aralığın dışındaki tüm veriler anında silinir.
- **İşitsel Maskeleme (Auditory Masking):** Eğer çok yüksek sesli bir davul (trampet) ve aynı anda çok kısık sesli bir fısıltı varsa, yüksek ses kısık sesi "maskeler". Bilgisayar o anki fısıltı verisini tamamen siler çünkü duyman imkansızdır.

2. Veri Sıkıştırma Neden Gereklidir?

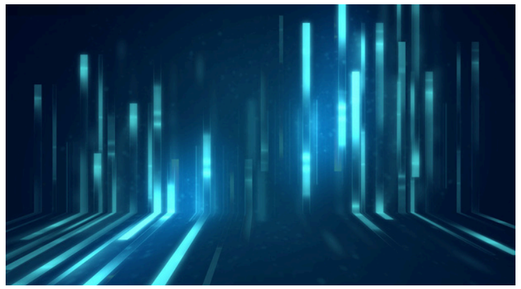
- Veri sıkıştırma, dijital dünyadaki verilerin (metin, resim, ses, video) içerdiği **gereksiz veya yinelenen bilgileri** ayıklayarak, verinin kapladığı alanı (bayt sayısını) küçültme işlemidir.

2.1 Depolama Alanından Tasarruf

- Günümüzde ürettiğimiz veri miktarı devasa boyutlara ulaştı. Sıkıştırma olmasaydı, cihazlarımızın kapasitesi çok çabuk dolardı.
- **Örnek:** Sıkıştırılmamış (RAW) bir fotoğraf yaklaşık 20-30 MB yer kaplarken, **JPEG** ile sıkıştırıldığında kaliteden çok az ödün vererek 2-3 MB'a düşebilir. Bu da telefonunuza 100 yerine 1000 fotoğraf sığdırmanızı sağlar.



2.2 İletim Hızı ve Bant Genişliği



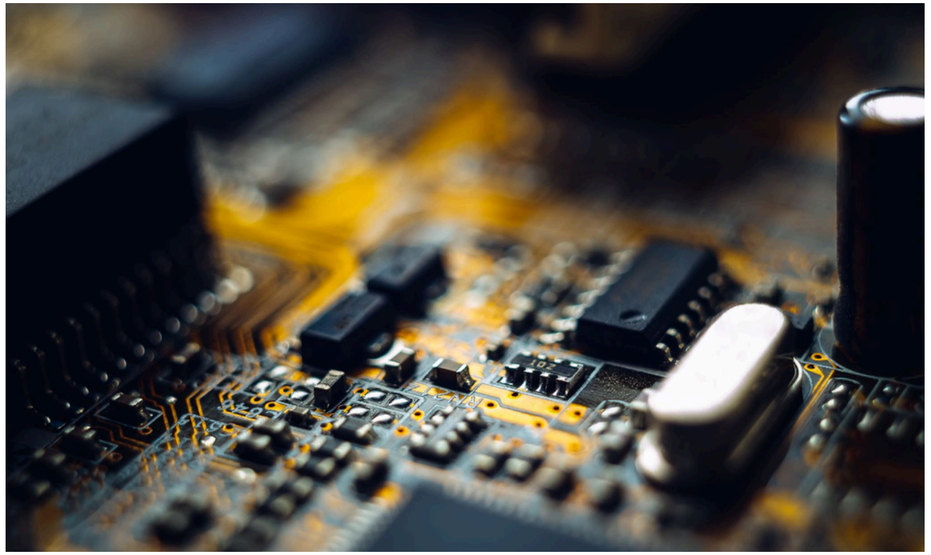
İnternet üzerinden bir veri gönderirken, dosya ne kadar küçükse karşı tarafa ulaşması o kadar hızlı olur.
Web Sayfaları: Tarayıcılar (Chrome, Safari vb.), web sitelerini indirirken arka planda sıkıştırılmış dosyalar kullanır. Bu sayede sayfalar saniyeler içinde açılır.
Video Akışı (Streaming): Netflix veya YouTube izlerken videolar anlık olarak sıkıştırılır. Sıkıştırma olmasaydı, 4K bir videoyu donmadan izlemek için bugünkünden 50 kat daha hızlı bir internete ihtiyaç duyardık.

2.3 Maliyetlerin Azaltılması

- Büyük şirketler (Google, Amazon, Meta) verilerini devasa sunucularda saklar.
- Veri miktarını %50 oranında sıkıştırmak, ihtiyaç duyulan sunucu sayısını ve harcanan **elektrik enerjisini** de yarı yarıya düşürür. Bu, hem ekonomik hem de çevresel bir kazançtır.

2.4 Donanım Performansı

- Küçük dosyalar, bilgisayarın belleği (RAM) ve işlemcisi arasında daha hızlı hareket eder. İşlemci, devasa bir veriyi yavaşça okumaktansa, sıkıştırılmış bir veriyi hızlıca çekip onu bellekte açmayı tercih eder.



3. Temel Kod Sıkıştırma Mantıkları/Yöntemleri

- **3.1 Run-Length Encoding (RLE) Mantığı:**

- **Çalışma Biçimi:** Veriyi tarar, aynı karakterin kaç kez arka arkaya geldiğini sayar ve bunu bir çift olarak kaydeder.

- Orijinal:AAAAABBBCCDAAAA(15 karakter)
- Sıkıştırılmış:5A3B2C1D4A(10 karakter)

- !!!!! Burada 15 karakterlik veri, 10 karaktere indirilmiştir. Ancak veride tekrar yoksa (örneğin ABCDE), RLE veriyi küçültmek yerine büyütebilir (1A1B1C1D1E).

3.2 Sözlük Tabanlı Sıkıştırma (LZ77 ve LZ78)

- Bu mantık, RLE’den bir adım daha ileri gider. Sadece yan yana duran tekrarları değil, metnin veya verinin genelinde **tekrar eden kalıpları** arar.
- Veriyi tararken daha önce karşılaştığı bir kelime veya dizi görürse, o diziyi tekrar yazmak yerine "X kadar geride, Y uzunluğunda bir dizi var" şeklinde bir referans (pointer) bırakır.

- **Nerede Kullanılır?** ZIP dosyaları, PNG görselleri ve HTTP sıkıştırmalarında (Gzip) yaygın olarak kullanılır.

3.3 Değişken Uzunluklu Kodlama (Huffman Coding)

- Bu yöntem, verinin içeriğine göre karakterlere farklı uzunlukta kodlar atar.
- Veri içinde **en çok kullanılan** karakterlere en kısa kodlar (örneğin 2 bit), **en az kullanılan** karakterlere ise en uzun kodlar (örneğin 10 bit) verilir.
- Toplamda harcanan bit sayısı önemli ölçüde azalır. Mors alfabesi bunun ilkel bir örneğidir (En çok kullanılan 'E' harfi sadece bir "nokta"dır).

ALGORİTMA	AVANTAJI	DEZAVANTAJI
RLE	Arka arkaya aynı piksellerin geldiği basit ikonlar, bitmapler.	Karışık metinlerde veriyi büyütebilir.
HUFFMAN	Her türlü metin dosyası veya genel veri.	Kodlamak için tüm veriyi önceden taraması gerekir.
LZ77(SÖZLÜK)	Kitaplar, kod dosyaları, loglar.	Bellek (RAM) kullanımı diğerlerine göre yüksektir.

Kayıplı vs. Kayıpsız Sıkıştırma

TÜR	MANTIK	ÖRNEKLER
Kayıpsız (Lossless)	Veri açıldığında orijinalin birebir aynısıdır. Hiçbir bit kaybolmaz.	RLE, ZIP, PNG, FLAC
Kayıplı (Lossy)	İnsan gözünün veya kulağının fark edemeyeceği detaylar silinir. Çok yüksek sıkıştırma sağlar.	JPEG, MP3, MP4

-Blok Bazlı Sıkıştırma:

- Görseller (özellikle JPEG) üzerindeki sıkıştırma mantığı, metin sıkıştırmadan çok daha farklı ve "akıllıca" bir yol izler. Burada amaç sadece tekrar eden veriyi bulmak değil, **insan gözünün kusurlarından faydalanmaktır.**
- JPEG gibi formatların kullandığı **Kayıplı Sıkıştırma** mantığının 3 temel aşamasını inceleyelim.

1. Renk Uzayı Dönüşümü (YCbCr)

- İnsan gözü, bir resimdeki **parlaklık** (ışık) değişimlerine karşı çok hassastır ama **renk** değişimlerine karşı o kadar duyarlı değildir.
  Bilgisayar resimleri genelde RGB(kırmızı,yeşil,mavi) olarak tutar.
  Sıkıştırma sırasında bu,Y(parlaklık),Cb/Cr(renk bileşenleri) olarak ikiye ayrılır.
  ** Renk verilerinin yarısını veya daha fazlasını atarız, ancak parlaklık verisine dokunmayız. Gözümüz bunu fark etmez bile!

2. Bloklama ve DCT (Ayrık Kosinüs Dönüşümü)

- Resim 8x8 piksellik küçük karelere bölünür. Her blok üzerinde matematiksel bir işlem olan **DCT** uygulanır.

- Bloktaki piksellerin tek tek değerlerini saklamak yerine, o bloktaki renk değişimlerinin "frekansını" saklarız.
- Sol üst köşe genel rengi (düşük frekans), sağ alt köşe ise ince detayları (yüksek frekans) temsil eder.

3. Niceleme (Quantization) - Asıl Sıkıştırma

- Bu adım JPEG'in "kayıplı" olmasının asıl sebebidir.
- İnsan gözü çok ince detaylardaki (yüksek frekanslı) değişimleri göremez.

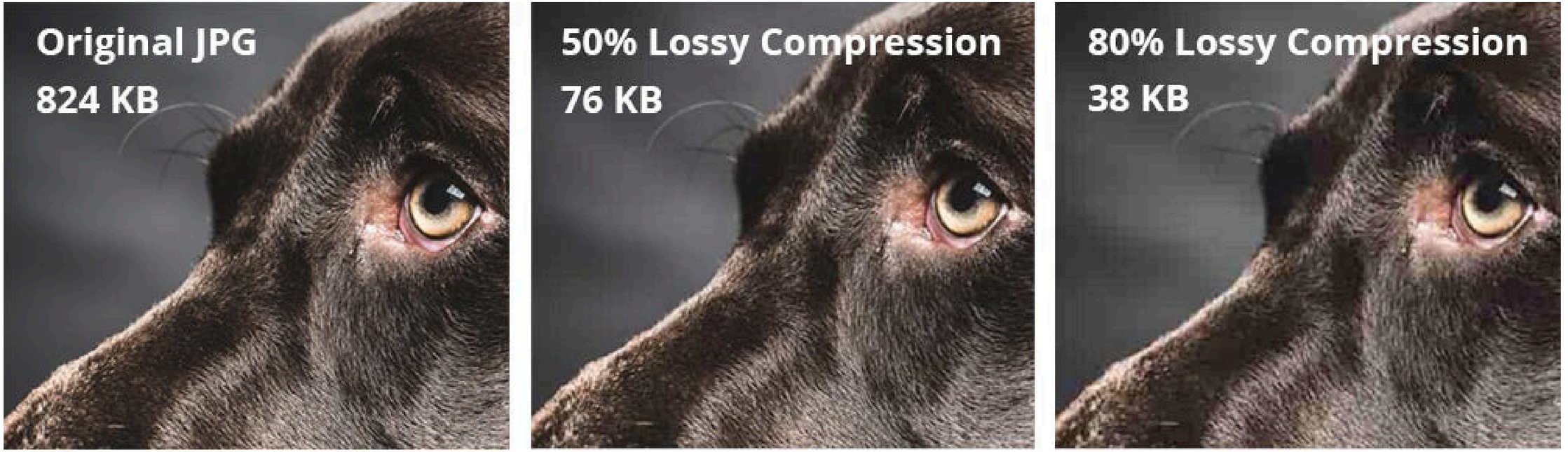
- Matematiksel bir tablo kullanılarak, sağ alt köşedeki (detay) değerler sıfıra yuvarlanır.

- Elimizde bolca **0** olan bir tablo kalır.

4. RLE ve Huffman Birleşimi

- Niceleme bittikten sonra tabloyu "zigzag" şeklinde okuruz. Bolca sıfır yan yana geldiği için RLE bu sıfırları müthiş bir hızla sıkıştırır. Üzerine bir de Huffman kodlaması eklenince, megabaytlarca yer kaplayan bir ham resim (RAW), kaliteden çok ödün vermeden kilobaytlara (JPEG) düşer.

- **Gözü Kandır:** Gereksiz renkleri at (YCbCr).
- **Matematiğe Dök:** Pikseli frekansa çevir (DCT).
- **Detayları Sil:** Önemsiz sayıları sıfırla (Niceleme).
- **Paketle:** Sıfırları RLE ve Huffman ile toparla.


Original JPG
824 KB
50% Lossy Compression
76 KB
80% Lossy Compression
38 KB

KAYNAKÇALAR:

- **Sayood, K. (2017).** *Introduction to Data Compression*.
- **D. A. (1952).** "A Method for the Construction of Minimum-Redundancy Codes". *Proceedings of the IRE*.
- Wikipedia.org
- ibm.com
- scienceDirect.com