

# VERİ DEPOLAMA VE SIKİSTİRMA ALGORİTMALARI

BİLGİSAYAR MÜHENDİSLİĞİNE GİRİŞ

SÜMEYYE ÇÖRDÜKÇÜ / 25360859091

# İÇİNDEKİLER

0) BİLGİSAYARIN VERİYİ  
GÖRME BİÇİMİ



1) METİN VERİLERİNİN BİT  
DÜZEYİNDE TEMSİLİ



2) GÖRSEL (RESİM)  
VERİLERİNİN BİT DÜZEYİNDE  
TEMSİLİ



3) SES VERİLERİNİN BİT  
DÜZEYİNDE TEMSİLİ



4) KARŞILAŞTIRMA



5) VERİ SIKIŞTIRMA



6) RLE GİBİ TEMEL  
SIKİŞTIRMA MANTIKLARI



# BİT DESENLERİ OLARAK BİLGİ GÖSTERİMİ

## O) BİLGİSAYARIN VERİYİ GÖRME BİÇİMİ

Bilgisayar biliminde bilgisayarların dış dünyadan aldığı her türlü bilgi -ister metin, ister görüntü, ister ses olsun- aslında yalnızca iki sembolden oluşan ikili bir alfabe ile, yani 0 ve 1 bitleriyle ifade edilir. İnsanlar için belirli anlamları olan sözcükler, renkli fotoğraflar veya şarkılar; bilgisayar belleğinde yalnızca elektriksel var-yok durumunu temsil eden bit dizileri şeklinde depolanır ve işlenir. Bu nedenle bilgisayarın bilgi işleme sürecini anlayabilmek için önce verinin bit seviyesinde nasıl soyutlandığını kavramak gereklidir.



# 1) METİN VERİLERİNİN BIT DÜZEYİNDE TEMSİLİ

Metinsel veriler insan dilinin sembolik yapısını yansıtır; ancak bilgisayar bu sembollerı harf ya da kelime olarak değil, sayısal kodlara dönüştürülmüş karakterler olarak yorumlar. Metindeki her karakter, bir karakter kodlama sistemi aracılığıyla sayıya çevrilir ve ardından bu sayılar ikili sisteme çevrilerek belleğe kaydedilir.



Bilgisayar bilimlerindeki en temel kodlama şeması ASCII'dir. Bu kod İngiliz alfabetesinin büyük ve küçük harflerini, noktalama işaretlerini, 0'dan 9'a kadar olan sayıları ve satır ilerletme, satır başı ve sekmeler gibi belirli kontrol bilgisini göstermek için 7 bit uzunlığında bit deseni kullanmaktadır. Örneğin; "A" harfinin ASCII'deki karşılığı 65'tir ve bellekte "01000001" şeklinde yer alır. Bunun nedeni sayılar ikili sisteminde gösterilirken sağdan başlayarak 2'nin 0.kuvvetinden artarak ilerler ve karşılık geldiği yerde 1 varsa eğer o değer toplanır. Burada da "A" karakteri ASCII'de 65 ile temsil ediliyordu.İkilik sistemdeki karşılığında da hesaplamızı yaparsak sonucun 65 geldiğini görürüz.



ASCII'den sonra ISO genişletilmiş ASCII standartları ortaya çıkmıştır.ISO her biri büyük bir dil grubu oluşturmak için tasarlanmış ASCII'ye ilave 128 desen geliştirmiştir.

Örneğin; İngiliz poundu ve Alman sesli harfleri için olan semboller de 128 ilave desenlerde bulunmaktadır.

ISO genişletilmiş ASCII standartları, dünyanın çok dilli iletişimini tamamını desteklemeye yönelik önemli bir ilerleme kaydetmiştir ancak iki önemli engel ortaya çıkmıştır.Birincisi: genişletilmiş ASCII'deki mevcut ekstra bit desenlerinin sayısı çoğu Asya ve bazı Doğu Avrupa dillerinin alfabelerini karşılamak için yetersiz kalmıştır.İkincisi: verilen bir belge sadece seçilen tek bir standarttaki semboller kullanmak ile kısıtlandırıldığı için, farklı dil gruplarından metinler içeren belgeler desteklenmemiştir.Bu eksikliği gidermek için Unicode ve UTF-8 gibi genişletilmiş kodlama sistemleri doğmuş; böylece dünya dillerine ait semboller 8,16 ya da 32 bitlik kodlarla temsil edilebilir hale gelmiştir.



## 2) GÖRSEL(RESİM) VERİLERİNİN BİT DÜZEYİNDE TEMSİLİ

Görüntüler, insan gözünde tek bir bütün olarak algılansa da bilgisayar açısından bir resim çok sayıdaki küçük hücrenin yanı piksellerin bir araya gelmesinden oluşan bir veri matrisi olarak yorumlanır. Her piksel belirli bir renk değerine sahiptir ve bu renkler sayısal biçimde kodlanır.

Bir bit eşlemde pikselleri kodlama yöntemi uygulamalara göre değişmektedir. Basit bir siyah-beyaz resimde, her piksel, ilgili pikselin siyah veya beyaz olmasına bağlı olarak aldığı değer ile tek bir bit tarafından gösterilebilmektedir. Bu en çok faks makineleri tarafından kullanılan yaklaşımdır. Renkli görüntülerde, her piksel daha karmaşık sistemler tarafından kodlanmaktadır.

Burada RGB kodlama karşımıza çıkar. Bunda her piksel üç renk bileşeni ile -kırmızı, yeşil, mavi-ışığın üç ana rengine karşılık gösterilmektedir. Bir byte her renk bileşeninin yoğunluğunu göstermek için kullanılmaktadır.



Bit eşlem görüntülerin en büyük dezavantajı, bir görüntüyü istenilen boyuta büyütüp küçültürken kalitenin bozulmasıdır. Çünkü görüntü tek tek piksellerden oluşur. Büyütülünce pikseller birer kare gibi görünür. Bu ölçekleme problemini önlemek için bir resmi piksel-piksel saklamak yerine onu matematiksel çizgiler eğriler, şekillerle tarif etmek mümkündür. Mesela bir daireyi piksellerle saklamak yerine; merkezi, yarıçapı gibi değerleri kaydedersin. Görüntüyü oluştururken bilgisayar bu matematiksel tanıma göre yeniden çizer. Bu yöntem sayesinde görüntü ister küçütlüsün ister büyütüsün çözünürlüğü bozulmaz, çünkü görüntü matematiksel olarak yeniden çizilir. Bu yüzden vektör grafikleri, yüksek kalite kaybı olmadan sınırsız ölçeklenebilir. Örneğin; TrueType metin sembollerini geometrik olarak tanımlamak için bir sistemdir. Aynı şekilde Post<script de metin ve resimlerin matematiksel tanımlarla kodlandığı bir teknolojidir. Bu yaklaşımlar özellikle CAD- bilgisayar destekli tasarım programlarında çok popülerdir. Çünkü mimarlar, mühendisler ürün tasarlarken şekillerin net, yeniden ölçeklenebilir ve doğru çizilmesine ihtiyaç duyar.

### 3) SES VERİLERİNİN BİT DÜZEYİNDE TEMSİLİ

Ses fiziksel dünyada hava moleküllerinin titreşimiyle oluşan analog (kesintisiz) bir olgudur; fakat bilgisayar bu kesintisiz yapıyı doğrudan saklayamaz. Bilgisayarlar görüntüyü piksel örnekleriyle sakladığı gibi sesi de örnekleyerek (sampling) saklar. Ses, analogdan dijitalle dönüştürme (A/D dönüşümü) süreciyle matematiksel bir dizi örnek değere dönüştürülür. Bilgisayar belirli aralıklarla ses dalgasından örnekler alır; bu işleme örneklemeye (sampling) adı verilir ve saniyede alınan örnek sayısı, örneklemeye oranını belirler.

Başka bir deyişle; ses dalgası kesintisiz bir yapıya sahiptir, bilgisayar ise sadece sayılar (bitler) ile çalışır. Bu nedenle ses, belirli aralıklarla genliği ölçüлerek sayısal değerlere dönüştürülür.

Örneğin metindeki sayı dizisi:

0, 1.5, 2.0, 1.5, 2.0, 3.0, 4.0, 3.0, 0

Bu değerlerin her biri, ses dalgasının o anki yüksekliğidir. Bilgisayar bu değerleri kaydeder daha sonra bu sayıları kullanarak sesin dalga şeklini yeniden oluşturabilir.

Örneklemeye Hızı:

Bu bir saniyede sesin kaç kez ölçüldüğü anlamına gelir.

Örneğin telefonlarda kullanılan örneklemeye oranı: 8000 örnek/sn. Yani saniyenin her 1/8000'lik aralığında ses bir kez ölçülür.

Ancak bu hız müzik için düşük kabul edilebilir, ses tam doğru ve kaliteli olmaz.

Müzik CD'lerinde: 44.100 örnek/sn.

Bu oran insan kulağının duyduğu ses kalitesine çok yakın bir kalite sağlar.

Yüksek örneklemeye hızı daha gerçekçi ve kaliteli ses demektir.

KURAL: Örneklemeye hızı ne kadar hızlı yapılrsa bilgisayarda ses o kadar doğru ve kaliteli temsil edilir ama dosya boyutu o kadar büyür.

## HER ÖRNEĞİN DİJİTAL KODLANMASI

Her ölçülen ses değeri, bilgisayar tarafından bit olarak saklanır.

Tek kanallı (mono) ses kayıtları genelde 16 bit olarak kaydedilir. Stereo ses kaydı için 32 bit gerekir.(çünkü iki kanal var = sol + sağ)

Sonuç; bir saniyelik stereo müzik milyonlarca bit bilgi demektir. Bu nedenle ses dosyaları disk üzerinde çok yer kaplar.

MIDI olarak bilinen bir alternatif kodlama sistemi bulunur.

### MIDI NASIL ÇALIŞIR?

Bir müzik enstrümanının çıkardığı ses dalgasını doğrudan örnekleyip kaydetmek yerine; hangi notanın çalındığı, ne kadar süreyle basıldığı, ses şiddeti gibi bilgileri kodlar.

Yani MIDI sesin kendisini saklamaz, müziğin talimatlarını saklar.

Örneğin;

"Piyanoda Do notası 2 saniye çalındı."

Bu bilgi bir milyon bitten fazla yer kaplamak yerine sadece 3-4 byte ile kodlanabilir.

Yani MIDI çok az yer kaplar çünkü sadece komutlar saklanır. Aynı zamanda MIDI, sesin kalitesinden çok müziğin nasıl çalındığını anlatmaya odaklanır. Bu nedenle gerçek müzik sesinin birebir kaydı değildir ama depolama açısından çok verimli bir yöntemdir.

# KARŞILAŞTIRMA

## METİN

Sembolik karakterlerin sayısal kodlanmasıyla dijital hale gelir.

## GÖRÜNTÜ

Piksellerin renk verisinin bit dizilerine çevrilmesiyle dijital hale gelir.

## SES

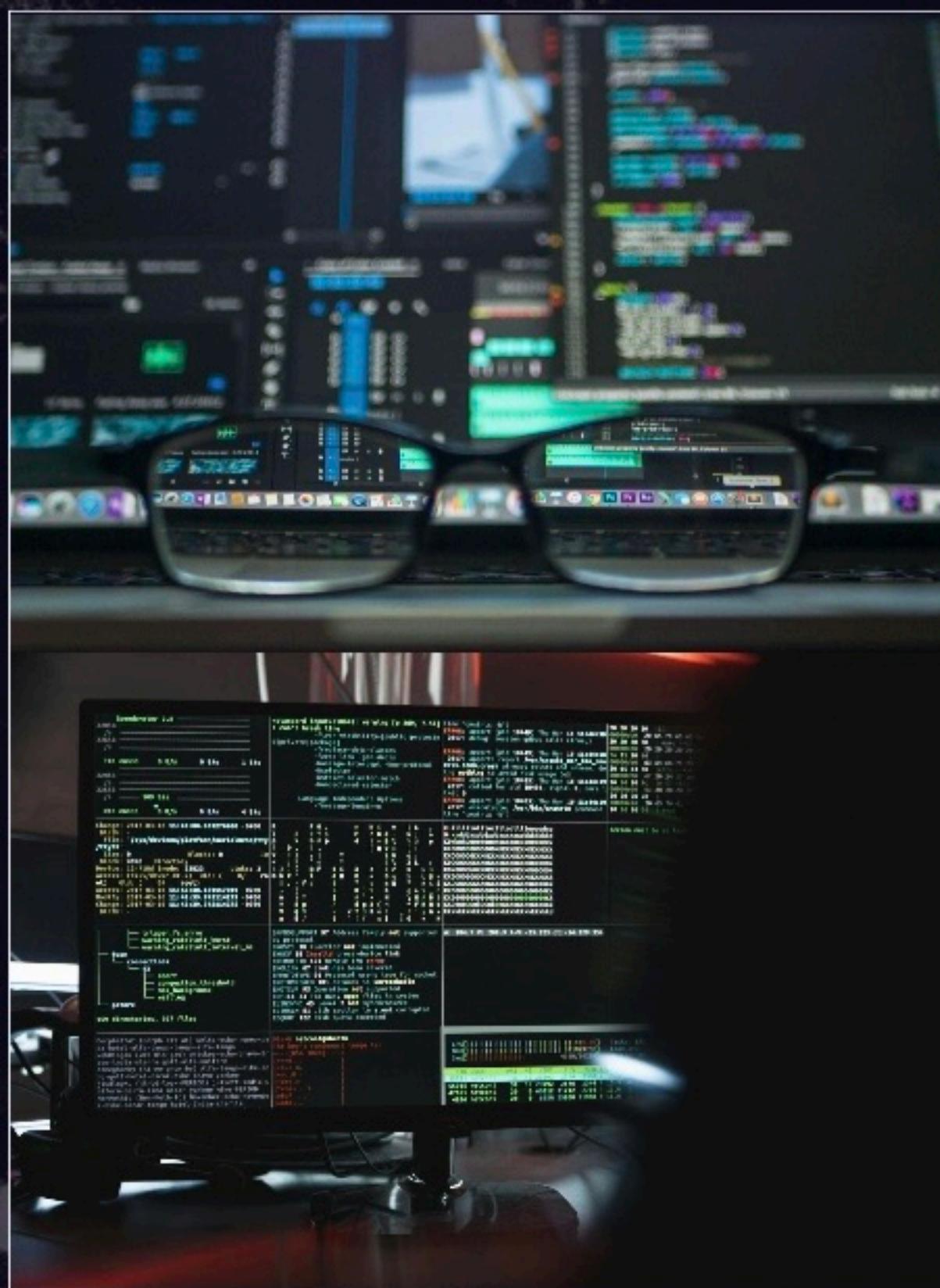
Analog (kesintisiz) dalganın örnekleme yoluyla sayısal değerlerine bölünmesiyle dijital hale gelir.

*Özetle; her veri türünün anlamı insan için farklı olsa da bilgisayar bu verilerin hepsini yalnızca farklı algoritmalarla işlenen ortak bir bit dili şeklinde algılar.*

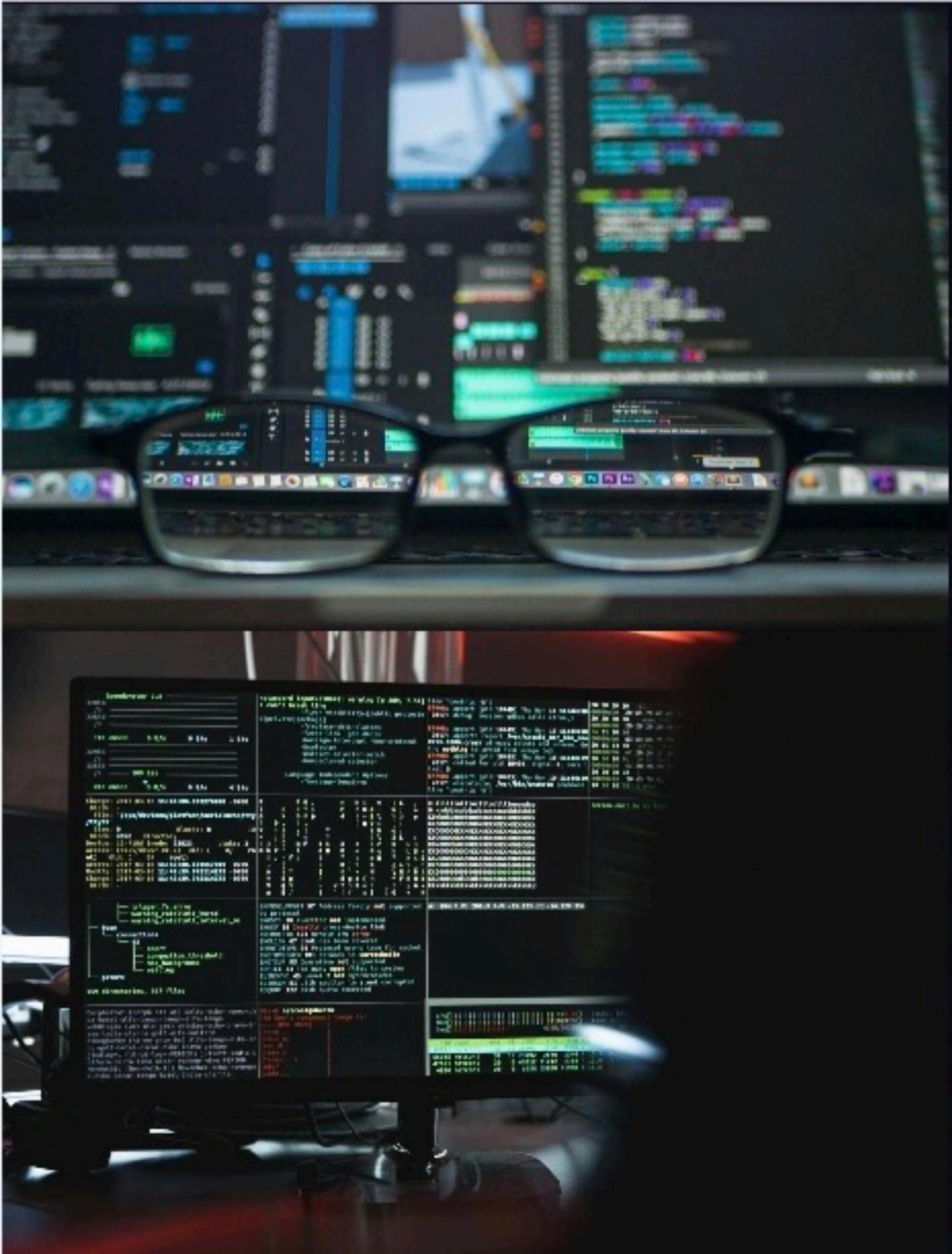
## 5) VERİ SIKIŞTIRMA

### VERİ SIKIŞTIRMA NEDİR?

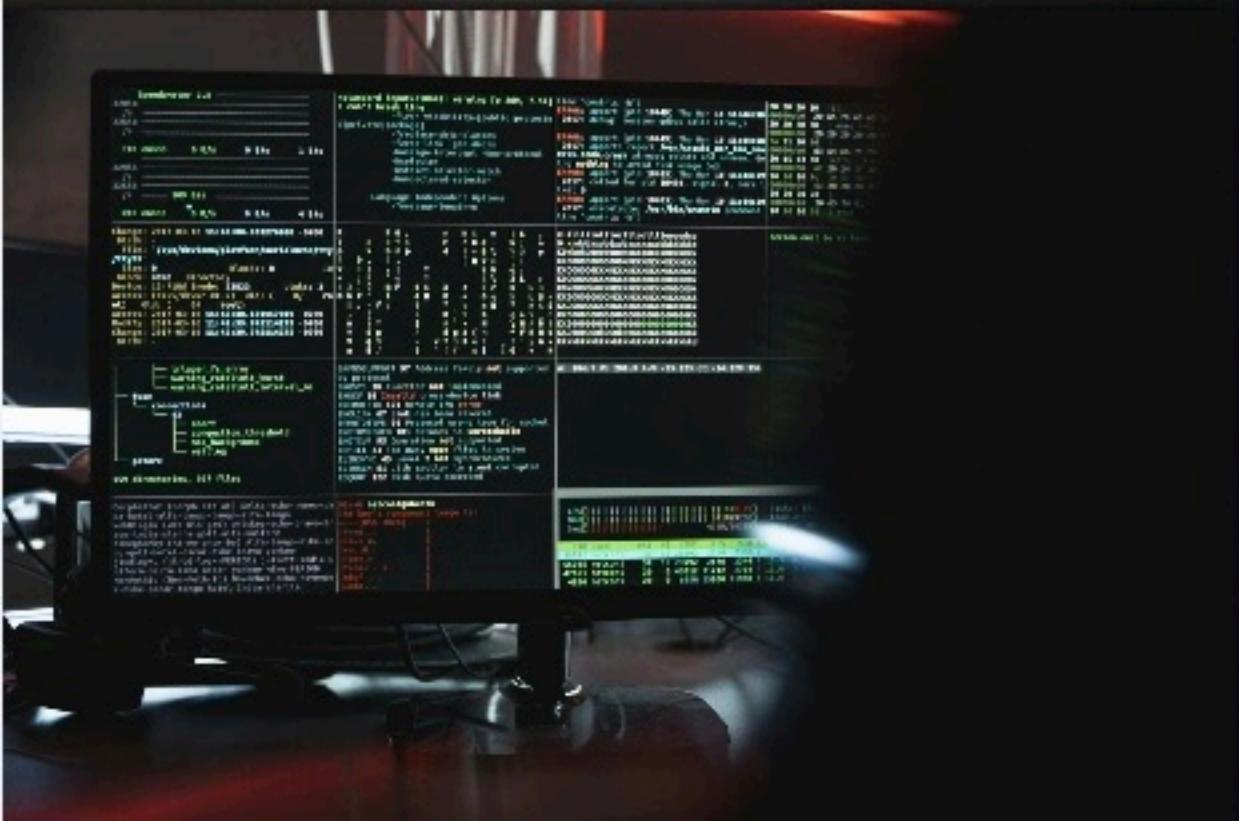
Veri sıkıştırma, bir verinin (metin, resim, ses, video, program dosyası vb.) kapladığı depolama alanını veya taşıma sırasında ihtiyaç duyduğu veri miktarını azaltma işlemidir. Yani veriyi daha az yer kaplayacak ve daha hızlı iletilecek hale getirmektir. Örneğin; normalde 1 GB yer kaplayan bir video, sıkıştırma işlemiyle 600 MB'a düşebilir. İçerik değişmez sadece daha verimli biçimde paketlenmiş olur.



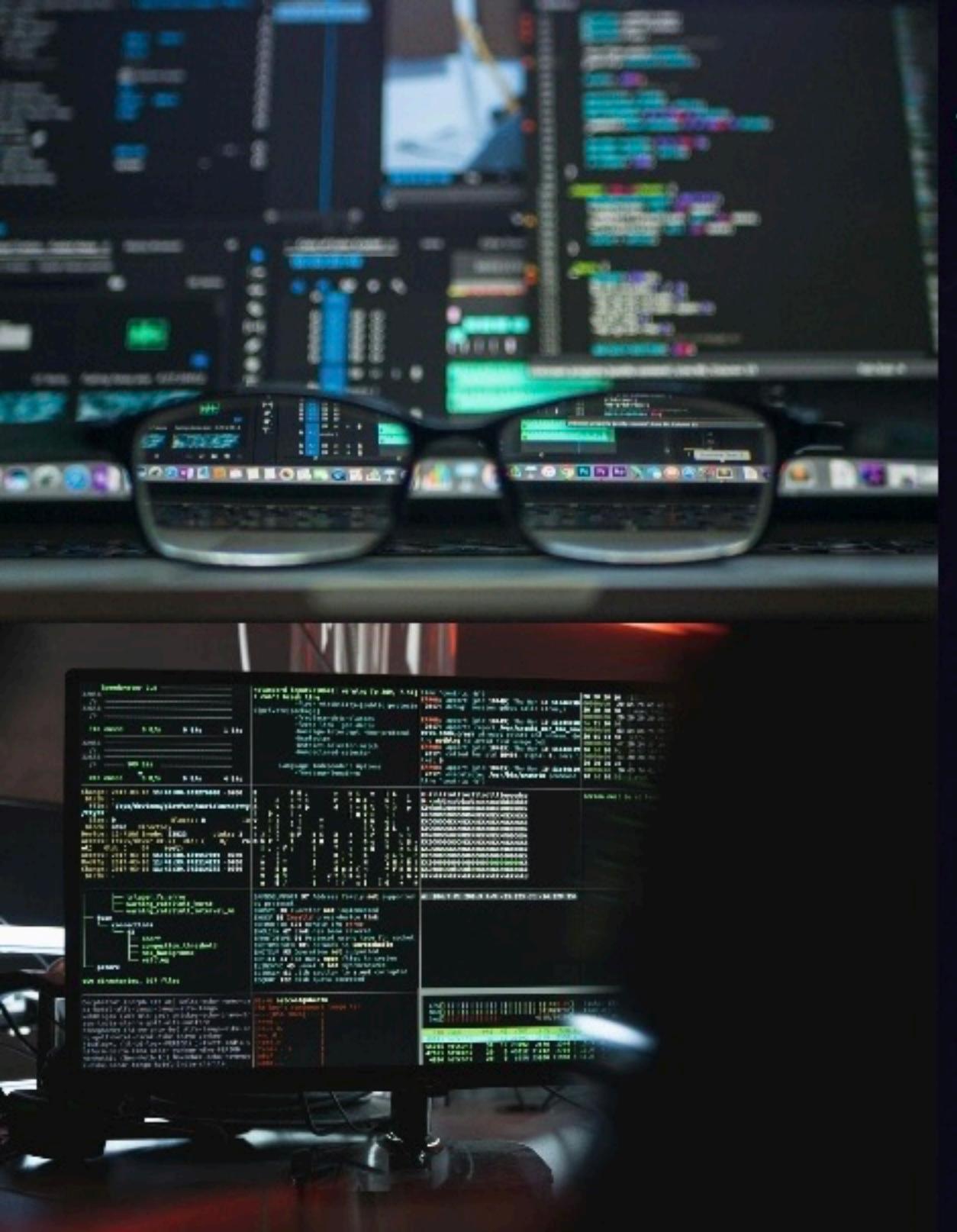
## GENEL VERİ SIKIŞTIRMA TEKNİKLERİ



Veri sıkıştırma şekilleri iki kategoriye ayrılmaktadır. Bazıları kayıpsız (lossless), bazıları ise kayıplıdır (lossy). Kayıpsız sıkıştırma ile sıkıştırılan bir veri tekrar açıldığında orijinal veri %100 aynı biçimde geri döner. Bu sıkıştırma türü metin dosyaları, programlar, veri tabanları, PNG gibi grafik formatlarında bilgi kaybının kabul edilemeyeceği durumlarda kullanılır.



Kayıplı sıkıştırma ise dosya boyutunu küçültmek için verinin insan tarafından fark edilmeyecek veya daha az önemli kabul edilen kısımlarını kalıcı olarak silen ve açma işleminden sonra orijinal haline tam olarak dönemeyen bir sıkıştırma türüdür. Kayıplı teknikler genellikle kayıpsız olanlardan daha çok sıkıştırma sağlarlar ve bu nedenle resim ve ses gibi küçük hataların tolere edilebileceği ayarlamalarda popülerdir.



# VERİ SIKIŞTIRMA NEDEN GEREKLİDİR?

## 1- DEPOLAMA ALANINI TASARRUFLU KULLANMAK İÇİN

Dijital sistemlerde veri saklamak maliyetlidir. Telefonumuzda, bilgisayarımızda hatta Google Drive gibi bulut servislerinde bile depolama alanı sınırlıdır.

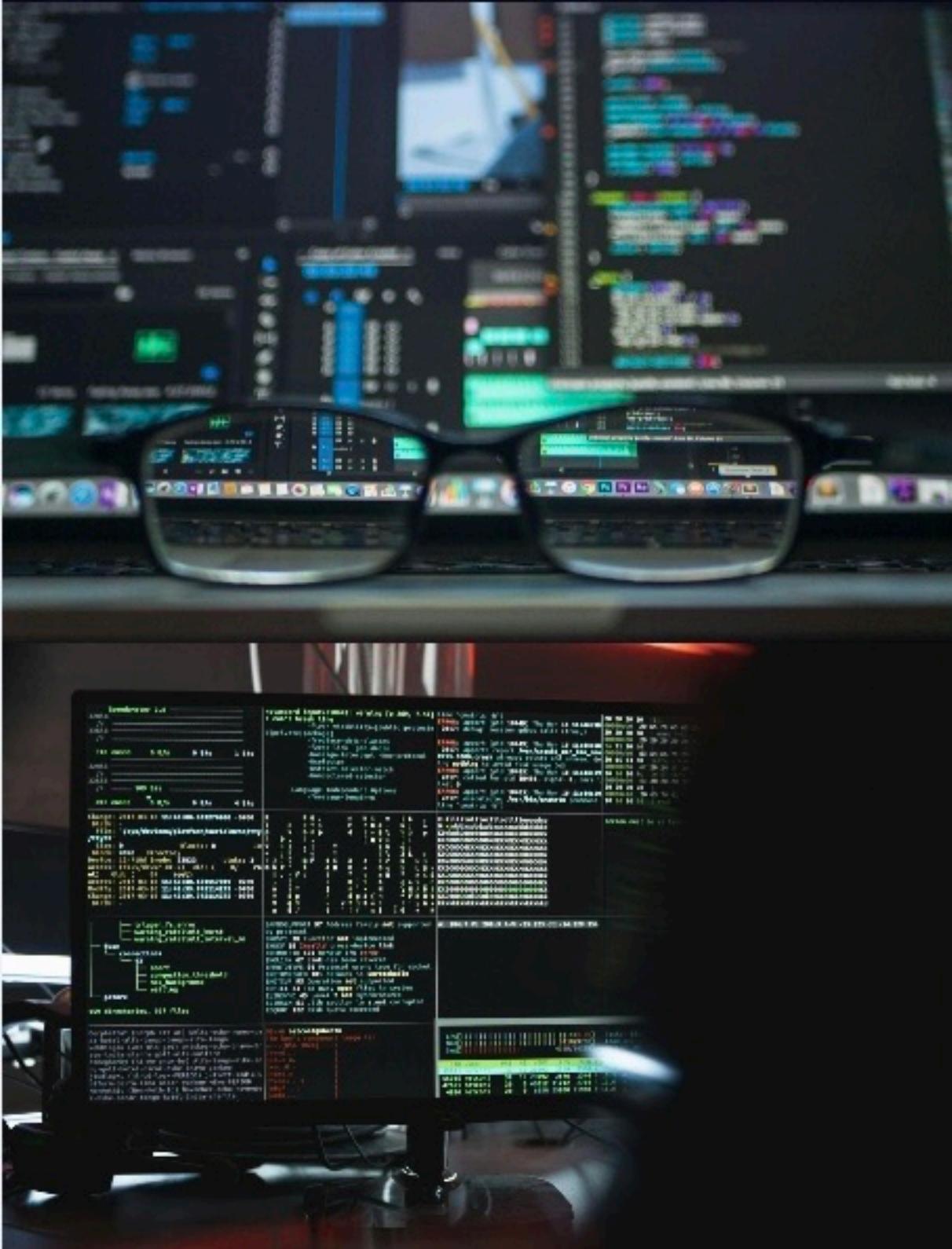
Eğer veri sıkıştırma olmasaydı:

- Bir cep telefonu yalnızca birkaç yüz fotoğraf alır sonra dolardı.
- Bir okulun öğrencileri için tek bir sunucu bile terabaytlarca yer tutardı.
- Netflix, YouTube gibi platformlar milyarlarca saatlik videoları depolayamazdı.

Örneğin sıkıştırma sayesinde;

- 3 MB'lık bir fotoğraf 400 KB'a düşebilir.
- 1 saatlik bir video 100 GB yerine 1-2 GB olabilir.

Bu sayede hem cihazlarda hem de sunucularda çok daha fazla veri saklamak mümkün olur.



## 2-VERİ AKTARIMINI VE İLETİŞİMİNİ HIZLANDIRMAK İÇİN

Veri yalnızca depolanmaz aynı zamanda taşınır.

Instagram'a fotoğraf yüklemek, WhatsApp'ta video göndermek, e-posta atmak... Hepsi veri transferi gerektirir.

Eğer verinin boyutu büyükse:

- Yükleme süresi uzar.
- İndirme süreler artar.
- İnternet bant genişliği dolar.
- Altyapı yavaşlar.

Bir fotoğrafı 300 KB yerine 30 KB göndermek, 10 kat daha hızlı yüklemek demektir.

Bugün WhatsApp'ta video izleyebiliyorsak, YouTube'da videolar donmadan akıyorsa bunun sebebi veri sıkıştırmanın veri akışını hızlandırmasıdır.

### 3 - MALİYETLERİ AZALTMAK İÇİN

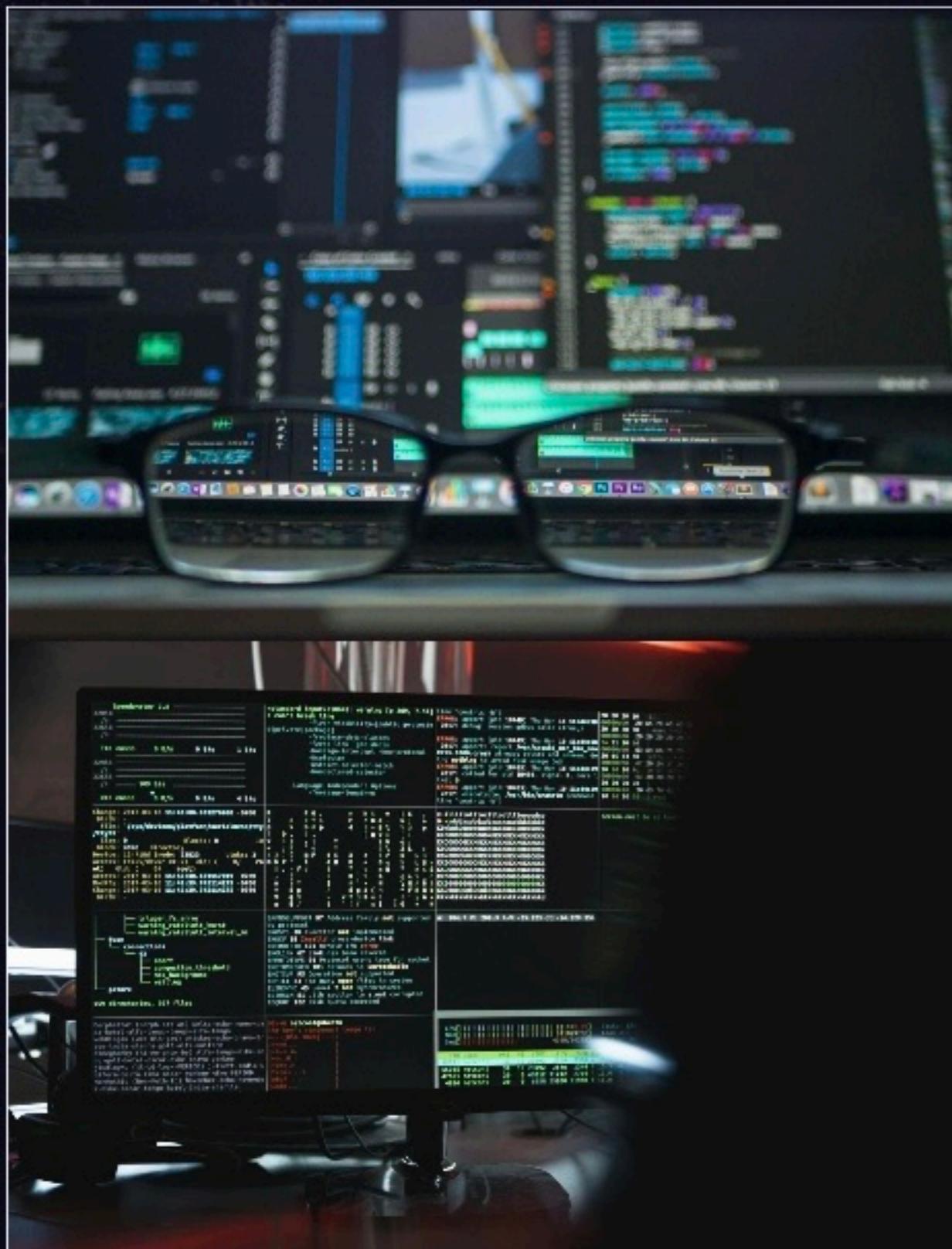
Veri saklamak da taşımak da ekonomik bir kaynağa bağlıdır.

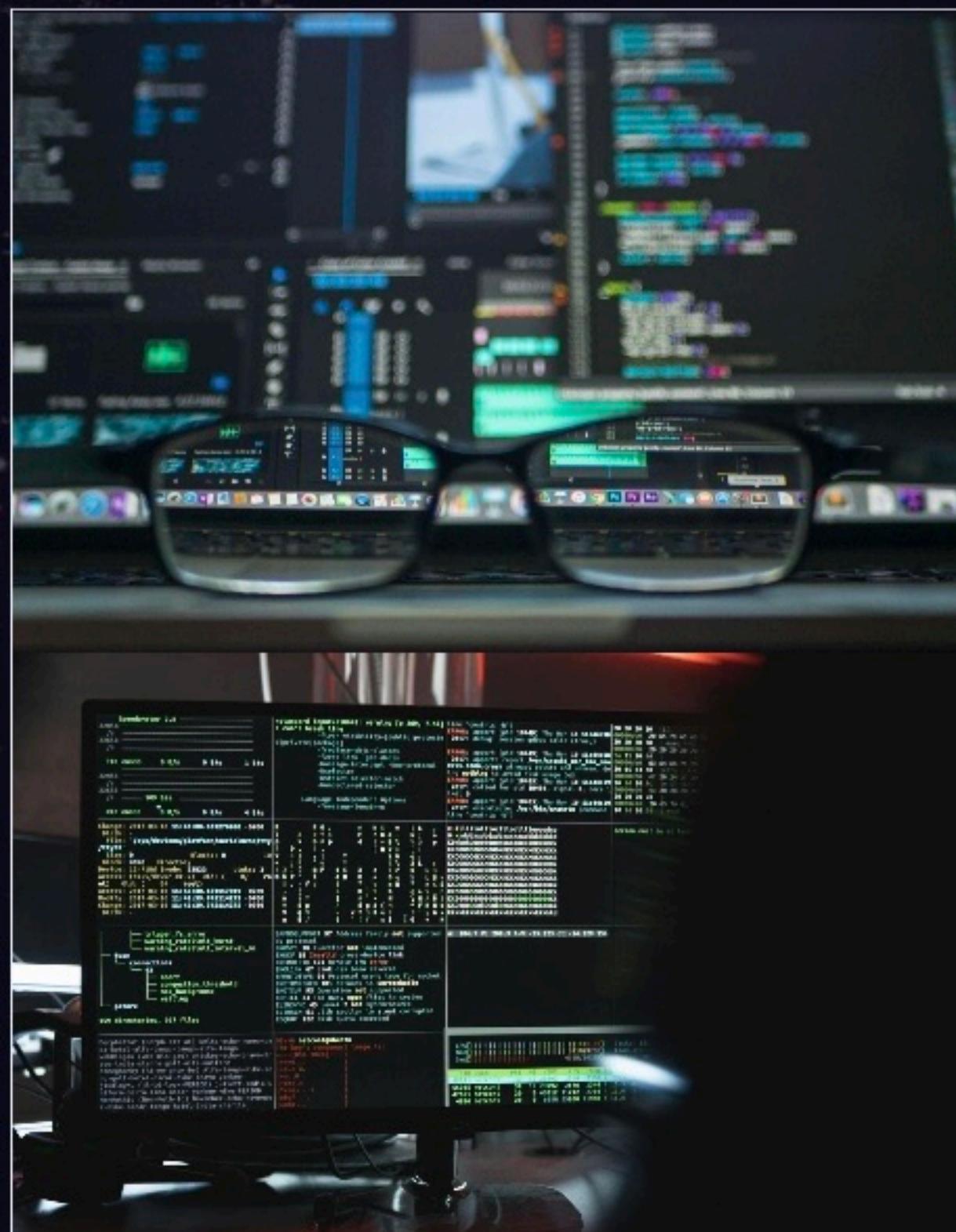
Bulut hizmetleri, sunucular, internet altyapısı şirketlere çok büyük maliyetlere mal olur.

Veri Sıkıştırma:

- Depolama ihtiyacını azaltır.
- Sunucu maliyetini giderir.
- Internet trafiğini azaltır.
- Enerji tüketimini azaltır.

Yani veri sıkıştırma yalnızca teknik bir konu değil, ekonomik sürdürülebilirliğin bir şartıdır.



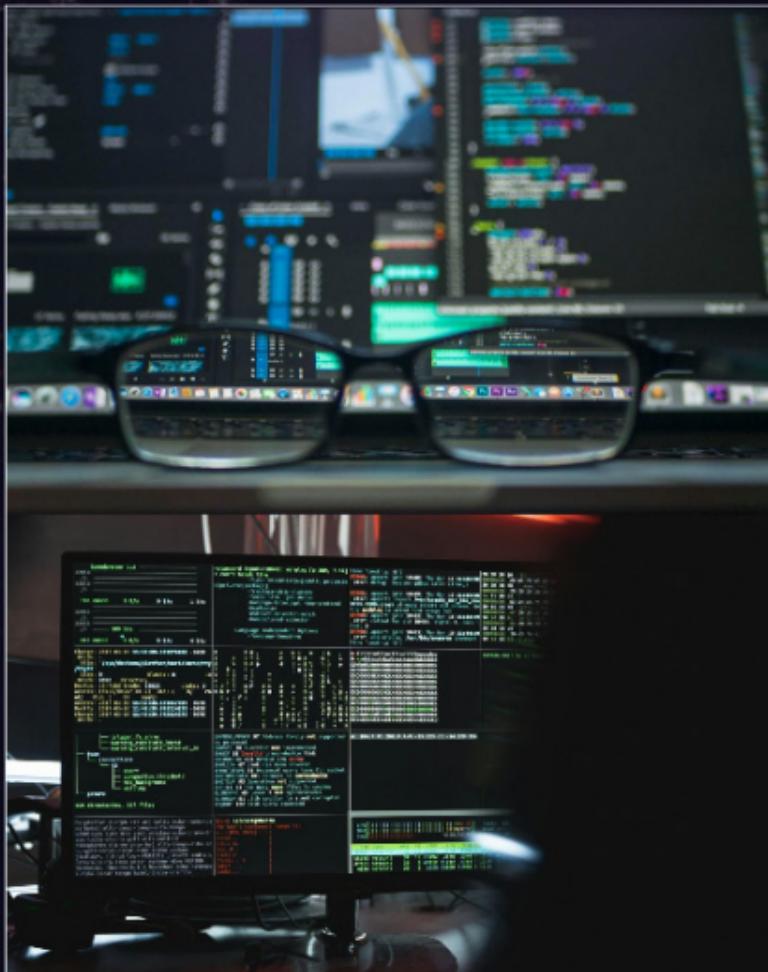


#### 4-SİSTEM PERFORMANSINI VE VERİMLİLİĞİNİ ARTIRMAK İÇİN

Küçük boyutlu veri:

- Belleğe daha hızlı yüklenir.
- CPU tarafından daha hızlı işlenir.
- Network üzerinde daha verimli akar.

Özellikle büyük sistemlerde örneğin; bankacılık sistemlerinde, e-ticaret platformlarında ve devlet veri tabanlarında milyonlarca kaydın saniyeler içinde işlenmesi gerekebilir. Bu işlemler veri sıkıştırılmadan neredeyse imkansızdır.



## 5 - TEKRARLAYAN VE GEREKSİZ BİLGİYİ AZALTMAK İÇİN

Veri içinde çoğu zaman aynı bilgi tekrar tekrar bulunur.

Örneğin;

AAAAAAABBBCCC

Bu 13 karakteri sıkıştırma algoritmalarıyla yalnızca:

7A3B3C

şeklinde temsil edebiliriz.

Bu tür mantıksal azaltmalarla bilgilere dokunmadan veri boyutu küçültülür.

Bu yöntem kayıpsız veri sıkıştırımıya örnektir.

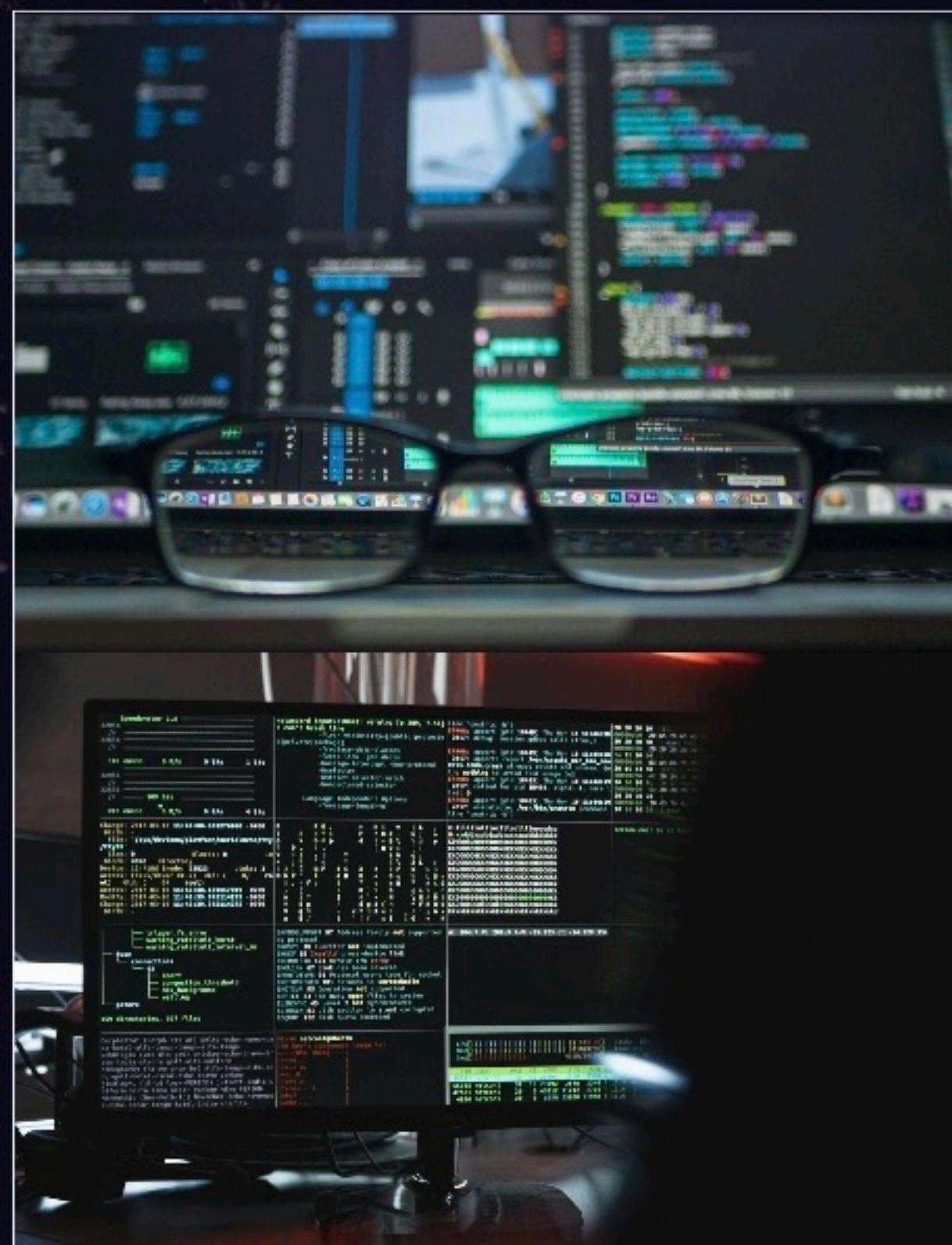
## 6 - MULTİMEDYA İÇERİĞİN KULLANILABILMESİ İÇİN (KAYIPLI SIKİŞTIRMA)

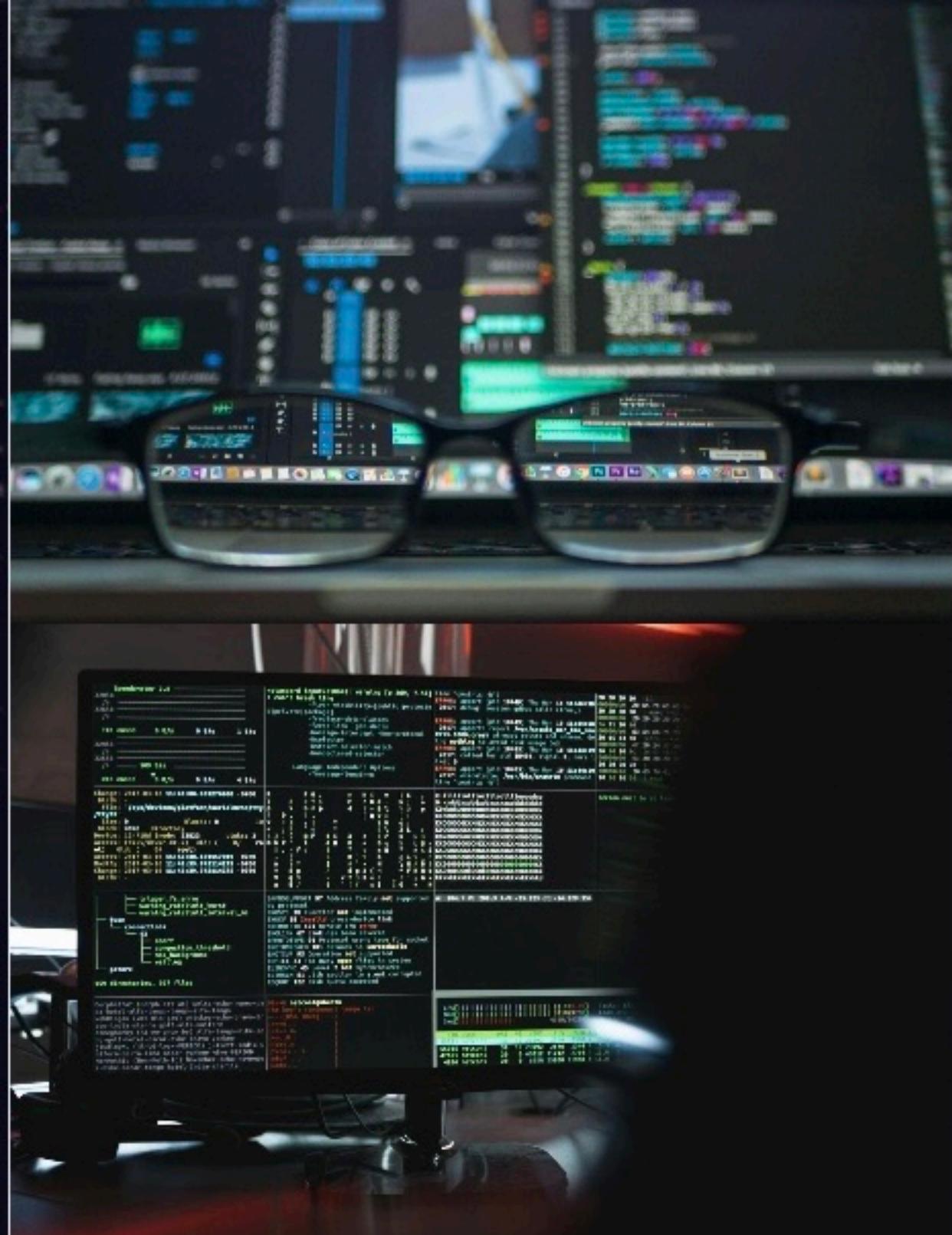
Bir dakikalık ham video kaydı gerçek haliyle yaklaşık 1-2 GB yer kaplar.

Bir dakikalık ham ses dosyası yüzlerce MB olabilir.

Bu veriler ham halleriyle internette saklanamaz veya oynatılamaz.

Bu nedenle JPEG, MP3, MP4 gibi kayıplı sıkıştırma teknikleri, insan gözünün ve kulağının fark edemeyeceği detayları atarak veri boyutunu %90-95 oranında azaltır.





# VERİ SIKIŞTIRMA OLMASAYDI NE OLURDU?

- Internetten video izlemek mümkün olmazdı.
- Telefonlarımızda birkaç dosya sonrası hafıza dolardı.
- Sosyal medya bugünkü haliyle var olamazdı.
- Veri merkezleri dünyaya yetmezdi.
- Bilgi paylaşımı çok yavaş, pahalı ve verimsiz olurdu.

# RUN-LENGTH ENCODING GİBİ TEMEL SIKIŞTIRMA MANTIKLARI

## a) Run - Length Encoding (RLE)

RLE, bilgisayar bilimlerinde kullanılan temel bir kayıpsız veri sıkıştırma tekniğidir. Bu yöntem, özellikle veri içerisinde aynı karakter veya değerlerin art arda tekrar ettiği durumlarda son derece etkili bir performans sergileyerek verinin temsil edilmesi için gereken bit miktarını azaltır.

## RLE'NİN AMACI NEDİR?

RLE'nin temel amacı, verinin yapısında tekrar eden öğeleri daha kısa bir ifade biçimiyile temsil etmek ve böylece veri depolama alanından tasarruf etmek ya da veri aktarımını hızlandırmaktır.



Çok basit bir örnek düşünelim:

AAAAAABBCCDAA

Bu dizide:

- A harfi art arda 5 kez
- B harfi 3 kez
- C harfi 2 kez
- D harfi 1 kez
- A harfi tekrar 2 kez

tekrar etmektedir.

RLE sıkıştırması bu veriyi şu biçimde dönüştürür:

5A3B2C1D2A

Buradaki fikir:

"AAAAAA" yazmak yerine "5A" yazmak çok daha az yer kaplar.



# RLE'NİN VERİ FORMATI MANTIĞI

RLE'nin mantığı sadece metin değil her tür veriye uygulanabilir. Yani RLE sadece harf sıkıştırırmaz, aynı zamanda:

- Bir resim verisindeki pikselleri
- Bir ses dosyasındaki aynı tondaki örnekleri
- Binary bit dizilerini

sıkıştırabilir.

Örneğin siyah-beyaz(0-1) bir görüntü satırı düşünelim:

00000000001111111111000

Bu satır RLE ile şöyle yazılabılır:

10\*0 10\*1 3\*0 veya 10 0 10 1 3 0

RLE burada pikseller arasındaki uzun tekrarları tespit ederek depolama maliyetini çok ciddi ölçüde düşürür.



# KODLAMA VE GERİ AÇMA MANTIĞI (ENCODING/DECODING)

Encoding (sıkıştırma) aşamasında veri analiz edilir ve sayı + karakter şeklinde yazılır.

Decoding (açma) aşamasında ise RLE çıktısı taranır ve sistem bu sayı-karakter çiftlerini tekrar orijinal hale genişletecek veriyi geri yükler.

Örnek:

4X2Y6Z

Decode İşlemi:

XXXXYYZZZZZZ



# SIKİŞTIRMA ORANI NASIL DEĞERLENDİRİLİR ?

Sıkıştırmanın ne kadar etkili olduğunu ölçmek için genelde şu formül kullanılır:

Sıkıştırma Oranı = Orijinal Veri Boyutu / Sıkıştırılmış Veri Boyutu

Oran 1'den büyükse sıkıştırma başarılıdır.

Örneğin;

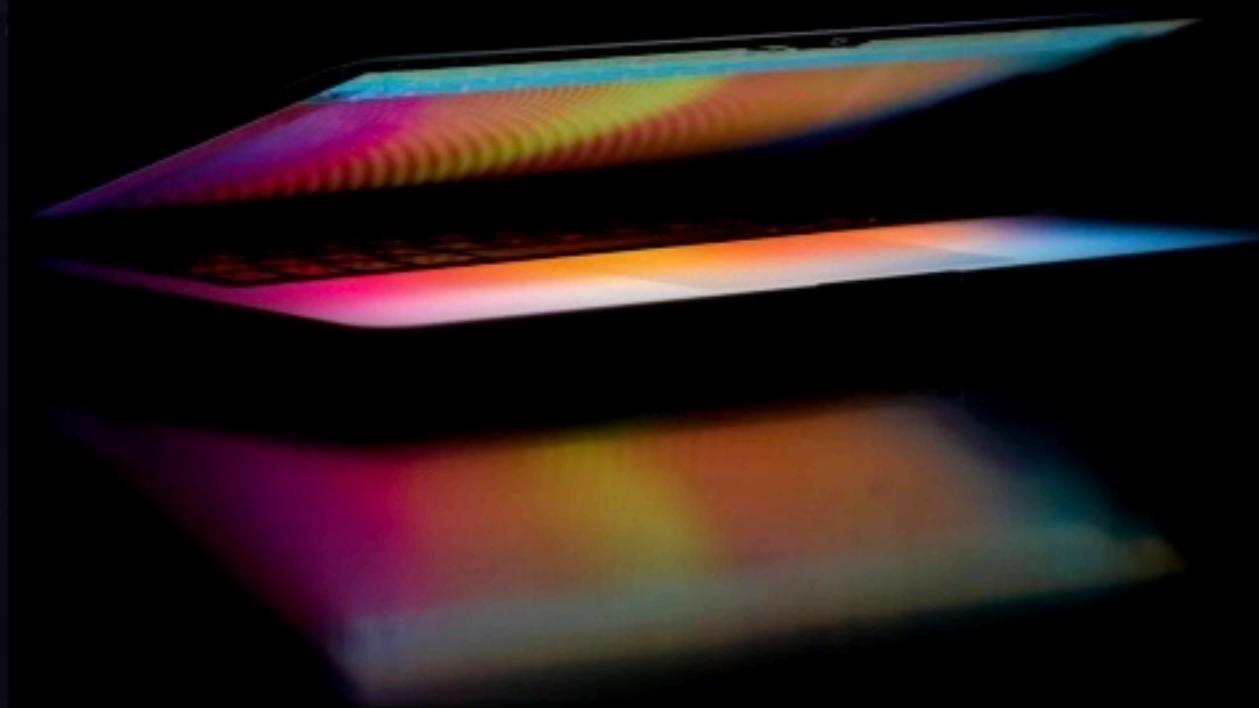
Orijinal = 12 karakter ( AAAAABBBCCDA )

Sıkıştırılmış = 10 karakter ( 5A3B2C1D2 )

Oran =  $12 / 10 = 1.2 \rightarrow \%20$  tasarruf



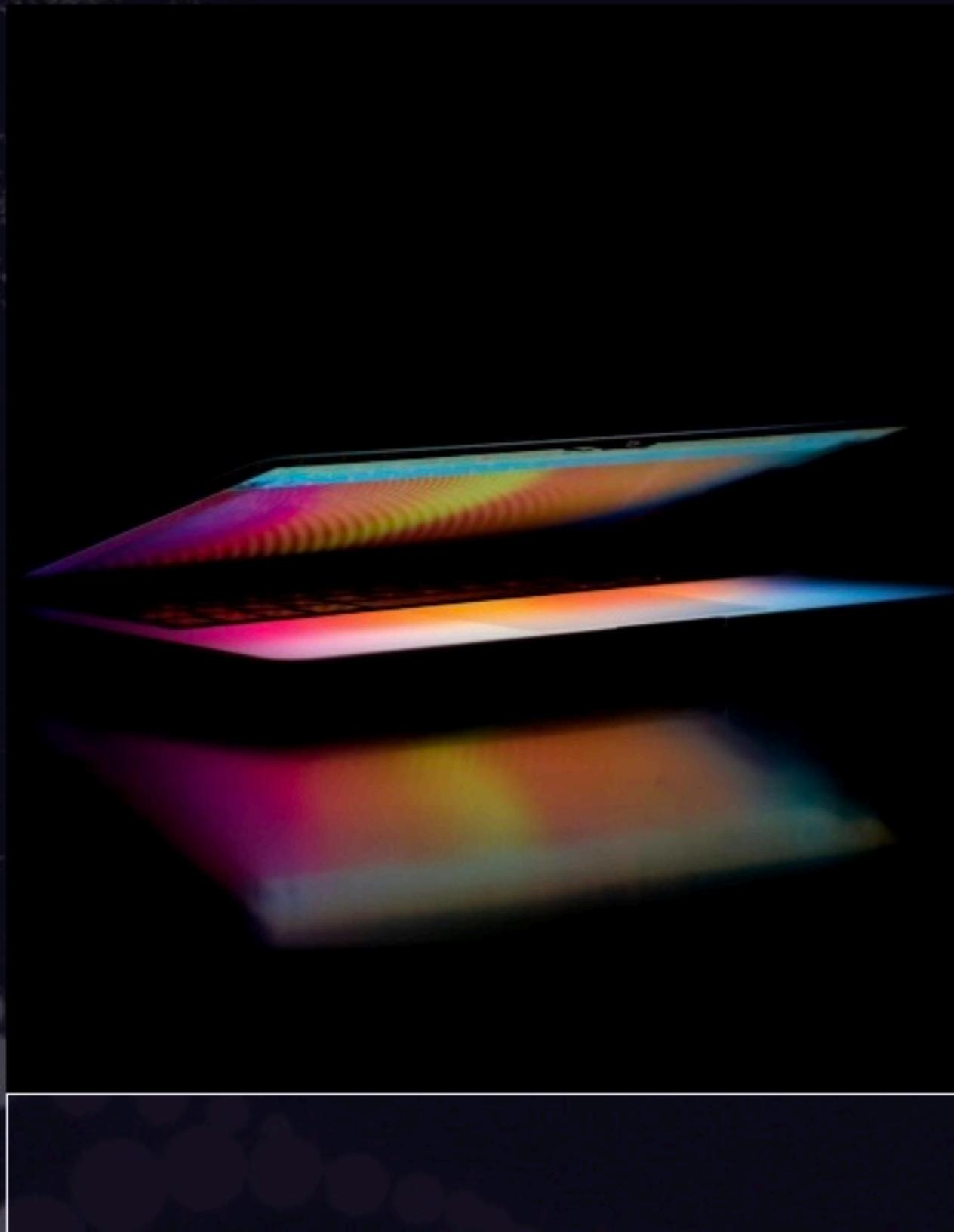
## b) Frekans Bağımlı Kodlama



Başka bir kayıpsız veri sıkıştırma tekniği, bir veri parçasını göstermek için kullanılan bit deseninin uzunluğunu o ögenin kullanım sıklığına ters olarak ilişkilendirdiği bir sistem olan frekans bağımlı kodlamadır. Bu tip kodlar, farklı uzunluklardaki desenler tarafından gösterilen öğeler anlamına gelen değişken-uzunluk kodlarına örnektirler. David Huffman, yaygın frekans bağımlı kodları geliştirmek için kullanılan bir algoritma keşfetmesi ile bilinmektedir ve bu amaç ile geliştirilen kodlara Huffman Kodları denir.

Frekans bağımlı kodlamanın bir örneği olarak, kodlanmış İngilizce dil metinlerinin görevini ele alalım. İngilizce dilinde e,t,a ve i harfleri z,q ve x harflerinden daha sık kullanılmaktadır. Bu nedenle İngilizce dilinde metin için bir kod oluşturulurken alan, eski harfleri göstermek için daha kısa bit desenleri; diğerlerini göstermek için daha uzun bit desenleri kullanarak kaydedilebilir.

Sonuç; İngilizce metnin eşit-uzunluk kodları ile elde edilenden daha kısa gösterime sahip olan bir kod olacaktır



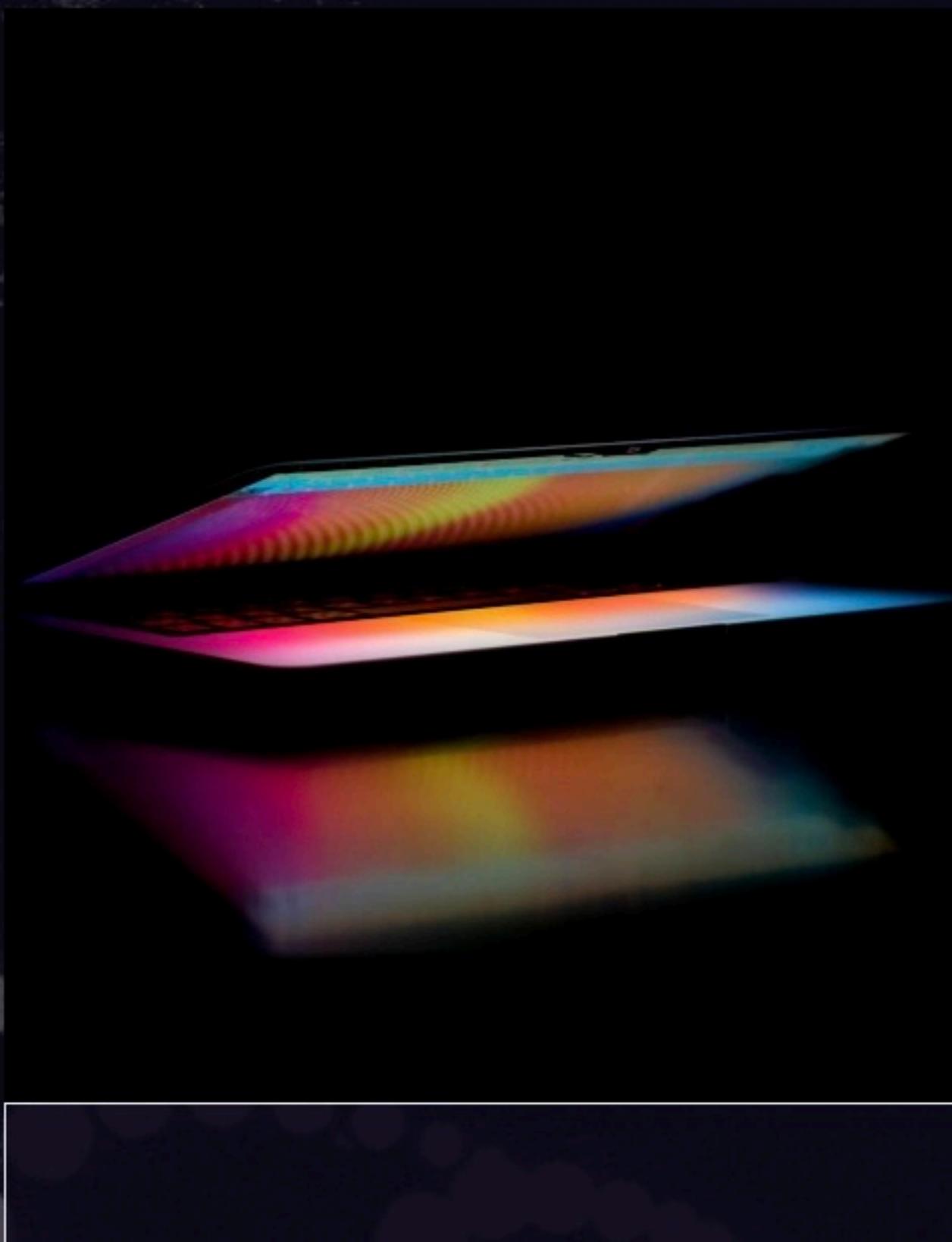
### c) Göreceli-Diferansiyel Kodlama

Bazı durumlarda, sıkıştırılacak veri, her biri bir öncekinden biraz farklı olan birimlerden oluşmaktadır. Örnek olarak, bir filmin art arda kareleri verilebilir. Bu durumlarda göreceli kodlama kullanan teknikler, diferansiyel kodlama olarak da bilinmektedir. Bu teknikler, tüm birimler yerine birbirini izleyen veri birimleri arasındaki farkları kayıt etmektedir, her birim bir önceki birim ile olan ilişkisi açısından kodlanmaktadır.

Göreceli kodlama, ardışık veri birimleri arasındaki farklılıklar tam olarak ya da yaklaşık olarak kodlanmasına bağlı olarak kayıplı ya da kayıpsız formda uygulanabilmektedir.

#### d) Sözlük Kodlama

Sözlük kodlama (dictionary encoding), veri sıkıştırma alanında kullanılan ve özellikle metin, resim ve benzeri tekrar eden bilgi gruplarının bulunduğu verilerde oldukça etkili olan bir yöntemdir. Bu teknikte amaç, veride sık kullanılan ifadeleri, kelimeleri ya da karakter dizilerini tekrar tekrar yazmak yerine, bunları tek bir kısa kod ile temsil ederek veri boyutunu küçültmektir. Sözlük kodlamada ilk olarak verinin genel yapısına bakılır ve veride çok sık tekrar edilen kelime, sembol veya veri parçaları belirlenir; ardından bunlar bir tabloya -yani bir sözlüğe- yerleştirilir. Daha sonra verinin içinde geçenbu ifadelerin her biri, sözlükteki kısa kodlarıyla değiştirilir. Örneğin bir metinde "the" kelimesi yüzlerce kez geçiyorsa, bu kelimenin her seferinde üç karakterlik yer kaplaması yerine, sözlükte "1" gibi tek bir rakam ile temsil edilmesi sağlanır; böylece tek bir sembol onlarca karakterlik bilginin yerine geçerek dosya boyutunu gözle görülür biçimde küçültür.



Sözlük kodlama, bazı sistemlerde önceden hazırlanmış sabit bir sözlük üzerinde çalışırken, daha gelişmiş bazı kodlayıcılar veriyi işlerken dinamik olarak yeni sözlükler oluşturur ve veri akışına göre kendini günceller. Bu yaklaşım sıkıştırma verimliliğini arttırmır çünkü sistem sık karşılaşılan yeni veri parçalarını otomatik olarak kaydeder ve sonraki tekrarlarında kısa kodlar kullanır. Sonuç olarak sözlük kodlama, verinin içeriğine bağlı olarak tekrar eden öğelerden faydalanan, hem depolama alanından tasarruf sağlayan hem de veri aktarımını hızlandıran etkili bir kayıpsız sıkıştırma tekniğidir.

**TEŞEKKÜRLER !**