JPEG Baseline Coding

How Your Photos Get Compressed

Mahesh C
FISAT

Federal Institute of Science and Technology (FISAT)
Multimedia Technology Class

February 3, 2026

Outline

A Simple Question

How big is a single photo from your phone?

A Simple Question

How big is a single photo from your phone?

Without compression:

- 12 MP camera = 12 million pixels
- Each pixel = 3 colors (RGB)
- Each color = 1 byte
- Total = 36 MB per photo!

With JPEG:

- Same 12 MP photo
- Only 2-4 MB
- 90% smaller!
- Still looks great

Real Life Impact

Think About This

Your phone has 128 GB storage. Without JPEG:

- You could store only **3,500 photos**
- With JPEG: **35,000+ photos**!

JPEG saves the day everywhere:

- **WhatsApp:** Sends photos in seconds, not minutes
- **Instagram:** Millions of photos uploaded daily
- **Websites:** Pages load fast with JPEG images
- **Email:** Attach photos without hitting size limits

What is JPEG?

JPEG = Joint Photographic Experts Group

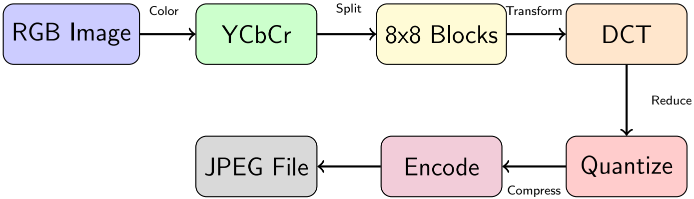
A committee that created the standard in 1992. Still the most popular image format today!

Key Facts:

- **Lossy compression** — throws away some data you won't notice
- **Best for photos** — not good for text, logos, or screenshots
- **Adjustable quality** — you choose size vs quality tradeoff
- **Universal** — works on every device, browser, app

"Good enough" quality at a fraction of the size

JPEG Compression: The Journey of a Photo



Six simple steps:

1. Change colors from RGB to YCbCr
2. Split image into small 8x8 pixel blocks
3. Transform each block (DCT magic)
4. Throw away details you won't see (Quantization)
5. Compress the numbers (Huffman coding)
6. Save as JPEG file

Think of it Like Packing for a Trip

Packing a suitcase:

1. Sort clothes by type
2. Fold into small bundles
3. Remove wrinkles (organize)
4. Leave behind things you won't need
5. Compress everything tight
6. Zip up the suitcase

JPEG compression:

1. Sort colors (RGB $\rightarrow$ YCbCr)
2. Split into 8x8 blocks
3. Organize data (DCT)
4. Remove invisible details
5. Compress numbers (Huffman)
6. Save as file

The Goal

Keep what matters, remove what doesn't, pack it efficiently!

Why Change Colors? RGB vs YCbCr

RGB (What cameras capture):

- Red, Green, Blue
- Each pixel = 3 values
- All three equally important
- Computer-friendly

Problem:
Can't compress any channel more than others — all seem equally important!

YCbCr (What JPEG uses):

- Y = Brightness (Luminance)
- Cb = Blue-ish color
- Cr = Red-ish color

Advantage:
Human eyes care more about brightness than color details!

The Human Eye Trick

Scientific Fact

Your eyes have 120 million cells for brightness but only 6 million for color!

What this means for JPEG:

- Keep full detail for brightness (Y)
- Reduce detail for colors (Cb, Cr) — you won't notice!
- This alone saves 50% of data

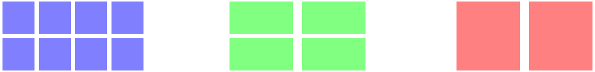
Real Example

A 1000x1000 pixel image:

- Y: Keep all 1,000,000 values
- Cb: Keep only 250,000 values (1/4)
- Cr: Keep only 250,000 values (1/4)

Chroma Subsampling: The Secret Sauce

4:4:4 (No reduction) **4:2:2 (Half horizontal)** **4:2:0 (Quarter)**



Full color info 50% color saved 75% color saved

Most JPEGs use 4:2:0:

- Every 2x2 block of pixels shares one color value
- You can't tell the difference in photos!
- Huge space savings with no visible quality loss

Why 8x8 Blocks?

The Idea:
Instead of processing the whole image at once, split it into tiny 8x8 pixel squares.

Why 8x8?

- Small enough to be similar inside
- Big enough to find patterns
- Perfect for the math that comes next
- Good balance of speed and quality

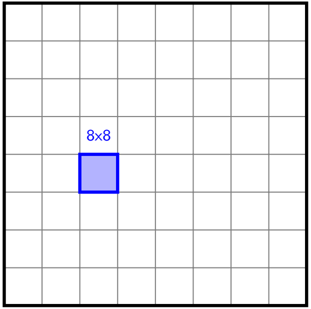
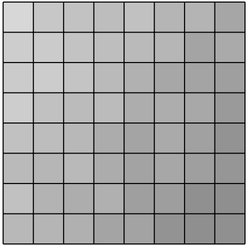


Image split into blocks

Example:
A 1920x1080 image = 32,400 blocks
Each processed independently!

What's Inside a Block?

A typical 8x8 block:



64 pixels, each with a brightness value (0-255)

Key observation:
Neighboring pixels are usually similar!

- Sky: all similar blue
- Skin: smooth gradients
- Grass: similar greens

This similarity = redundancy
Redundancy = opportunity to compress!

DCT: Don't Worry About the Math!

What DCT Does (Simple Version)

Converts pixel values into "frequency" information.
Think of it as finding patterns in the block.

Analogy: Music Equalizer

- Music has bass (low frequency) and treble (high frequency)
- An equalizer shows how much of each
- DCT does the same for images!

Low frequency =

- Smooth areas
- Gradual changes
- The "big picture"

High frequency =

- Sharp edges
- Fine details
- Texture and noise

Before and After DCT

Before DCT:

64 pixel values scattered all over

102	145	110	141	116	107	136	175
172	152	127	175	120	105	114	141
162	184	149	103	135	151	132	161
185	147	100	149	122	126	174	175
118	154	137	132	132	143	113	151
186	190	169	172	121	175	136	123
172	130	112	103	155	135	105	104
106	159	118	145	131	194	165	147

All values seem important

After DCT:

Energy concentrated in corner!

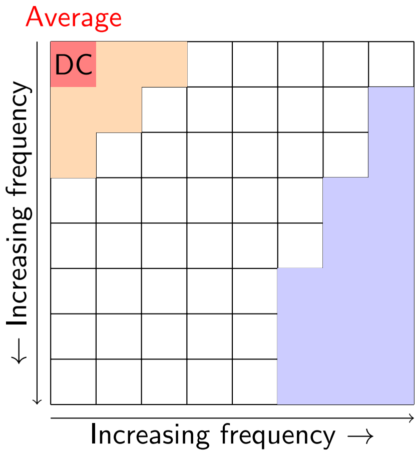
952	-24	12	0	0	0	0	0
-18	5						
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

Most values are zero or tiny!

The Magic

DCT doesn't compress anything yet — it just reorganizes data so compression becomes easy!

The DCT Coefficient Map



- **Top-left (DC):** Average brightness of the block
- **Near top-left:** Low frequency = smooth patterns (important!)

Quantization: The Lossy Step

This is where JPEG becomes "lossy"

Quantization throws away information you (hopefully) won't miss!

Simple Analogy: Rounding Money

- You have \$47.83
- Round to nearest \$10 $\rightarrow$ \$50
- You lost \$2.17 of precision
- But for rough estimates, \$50 is "good enough"

JPEG does the same:

- DCT gives precise values like 47.83
- Quantization rounds them: 47.83 $\rightarrow$ 5
- Less precise, but much smaller numbers!

The Quantization Table

Divide each DCT value by a number:

16	11	10	16				
12	12	14	19				
14	13	16	24				
						99	99
						99	99

Small divisors (top-left) = keep detail
Big divisors (bottom-right) = lose detail

Example:
DCT value = 95
Quantization divisor = 16
Result = round$(95/16)$ = **6**

High frequency example:
DCT value = 12
Quantization divisor = 99
Result = round$(12/99)$ = **0**

Small details become zero!

Quality Settings Explained

What does "JPEG Quality 80%" mean?

Quality 100%

- Small divisors
- Keep most detail
- Larger file
- Best quality

Quality 75%

- Medium divisors
- Good balance
- Reasonable size
- *Most common*

Quality 20%

- Large divisors
- Lose lots of detail
- Tiny file
- Visible artifacts

Rule of Thumb

Quality 70-85% is usually the sweet spot — good quality, reasonable size.

What Happens at Low Quality?

JPEG Artifacts — Signs of Over-Compression:

1. Blocking

- 8x8 block boundaries become visible
- Smooth areas look "blocky"
- Like a mosaic effect

2. Ringing

- Halos around sharp edges
- "Ghosting" near text
- Wavy patterns

3. Color Bleeding

- Colors smear into each other
- Red bleeds into white
- Especially around edges

4. Mosquito Noise

- Fuzzy dots around edges
- Looks like tiny insects
- Common in video too

After Quantization: Lots of Zeros!

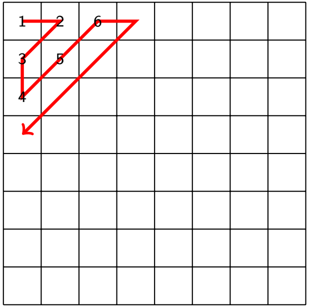
A typical quantized block:

42	-3	2	-1	0	0	0	0
-5	1	0	0	0	0	0	0
2	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

Notice: Most values are 0! Only a few non-zero values in the top-left.
Opportunity: Instead of storing 64 numbers, just store the non-zero ones!

Zigzag Scan: Reading the Block

Read in zigzag order:



Why zigzag?

- Groups low frequencies first
- High frequencies (zeros) come last
- Long runs of zeros at the end
- Easy to compress!

Result:

42, -3, -5, 2, 1, 2, -1, 0, 0, 0, 0, 0, ... (lots of zeros)

Run-Length and Huffman Coding

Two tricks to compress the zigzag sequence:

1. Run-Length Encoding

Instead of: 5, 0, 0, 0, 0, 3
Write: 5, (4 zeros), 3
Or even shorter: (0,5), (4,3)

"Skip 0 zeros, value is 5"
"Skip 4 zeros, value is 3"

2. Huffman Coding

Common patterns get short codes:

- "End of block" = very short
- Small values = short codes
- Rare values = longer codes

Remember from last class!

Combined Effect

A block with 64 values might compress to just 20-30 bits!

DC Coefficient: Special Treatment

The DC coefficient (average brightness) is special:

Observation:
Neighboring blocks have similar average brightness.

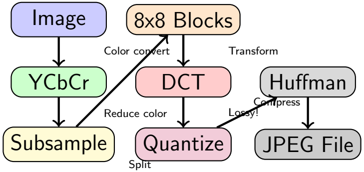
Block 1 DC: 125
Block 2 DC: 128
Block 3 DC: 130
Block 4 DC: 127

DPCM Trick:
Store differences instead!

Block 1: 125 (first one)
Block 2: +3 (128-125)
Block 3: +2 (130-128)
Block 4: -3 (127-130)

Small numbers = better compression!

The Complete JPEG Pipeline



Summary of each step:

1. **YCbCr:** Separate brightness from color
2. **Subsample:** Reduce color resolution (humans won't notice)
3. **8x8 Blocks:** Divide and conquer
4. **DCT:** Find patterns, concentrate energy
5. **Quantize:** Round aggressively (this is where quality is lost)
6. **Huffman:** Compress the numbers efficiently

Decoding: The Reverse Journey

To view a JPEG, reverse all steps:

JPEG File → Huffman Decod → Dequantize → Inverse DCT → RGB Image

Important!

Decoding cannot recover the original image perfectly!
The information lost during quantization is gone forever.
That's why JPEG is called "lossy" compression.

Scenario 1: Sharing Photos on WhatsApp

What happens when you send a photo:

1. You take a 12 MP photo (36 MB raw)
2. Phone saves as JPEG (3 MB)
3. WhatsApp re-compresses to 100 KB
4. Friend receives smaller image

Why so aggressive?

- Faster upload/download
- Less mobile data used
- Works on slow connections

The tradeoff:

Original: 4000 x 3000 pixels
WhatsApp: 1280 x 960 pixels
Quality: 70%

Pro Tip

Send as "Document" to preserve original quality!

Scenario 2: Website Images

Why website images need careful optimization:

Page load time matters:

- 1 second delay = 7% fewer sales
- Google ranks faster sites higher
- Users leave slow sites

Typical website image:

- Hero image: 200-400 KB
- Thumbnails: 10-30 KB
- Icons: Use PNG/SVG instead

Best practices:

- Quality 70-85% for photos
- Resize to actual display size
- Use responsive images
- Consider WebP format

Common Mistake

Uploading 5 MB photos that display at 500x300 pixels!

Scenario 3: Professional Photography

When quality matters most:

Workflow:

1. Shoot in RAW format
2. Edit in Lightroom/Photoshop
3. Export as high-quality JPEG
4. Keep RAW as backup

Export settings:

- Quality: 90-100%
- Color space: sRGB for web
- Resolution: Full size

Why not just use RAW?

- RAW files are huge (25-50 MB)
- Not universally supported
- Can't share easily
- JPEG is "good enough" for delivery

Pro Tip

Never edit and re-save JPEG multiple times — quality degrades each time!

Scenario 4: Medical Imaging

Critical Application

Medical images (X-rays, MRIs) often use **lossless** formats, not JPEG!

Why JPEG can be dangerous for medical images:

- Small details might indicate disease
- Compression artifacts could hide tumors
- Legal requirements for image integrity
- Diagnosis accuracy is critical

What they use instead:

- DICOM format (medical standard)
- Lossless JPEG (yes, it exists!)
- JPEG 2000 with lossless mode
- PNG for some applications

Scenario 5: Social Media Memes

Why do memes look so bad?

Download $\rightarrow$ Edit $\rightarrow$ Upload $\rightarrow$ Download $\rightarrow$ Edit $\rightarrow$ Upload...

Generation Loss:

- Each save loses quality
- Platforms re-compress
- Text becomes unreadable
- Colors get muddy
- Artifacts multiply

After 10+ generations:

- Severe blocking
- Color banding
- Blurry edges
- "Deep fried" look

This is actually used as an aesthetic in "deep fried memes"!

When to Use JPEG

JPEG is GREAT for:

- Photographs
- Natural images
- Complex scenes
- Gradients and shadows
- Web photos
- Social media

JPEG is BAD for:

- Text and logos
- Screenshots
- Line art
- Images with transparency
- Graphics with sharp edges
- Images needing editing

Simple Rule

Photo? → JPEG **Graphics/Text?** → PNG

Format Comparison

Feature	JPEG	PNG	WebP	HEIC
Compression	Lossy	Lossless	Both	Both
Transparency	No	Yes	Yes	Yes
Animation	No	No	Yes	Yes
File Size	Small	Large	Smaller	Smallest
Quality	Good	Perfect	Better	Best
Support	Universal	Universal	Good	Apple

The future:

- WebP is replacing JPEG on the web
- HEIC is default on iPhones
- AVIF is the newest contender
- But JPEG will be around for decades!

Key Takeaways

1. **JPEG exploits human vision** — we don't see all details equally
2. **Six-step pipeline:** Color convert → Subsample → Block → DCT → Quantize → Encode
3. **Quantization is the lossy step** — this is where quality vs size tradeoff happens
4. **Quality 70-85%** is usually the sweet spot
5. **Don't re-save JPEGs** — quality degrades each time
6. **Use JPEG for photos**, PNG for graphics

Quick Reference

JPEG Quality Guide

- **100%:** Maximum quality, large files (archival)
- **90-95%:** Excellent quality, professional use
- **75-85%:** Good quality, web/social media
- **50-70%:** Acceptable, thumbnails
- **Below 50%:** Visible artifacts, avoid

File Size Estimates (12 MP photo)

- RAW: 25-50 MB
- JPEG 100%: 8-12 MB
- JPEG 80%: 2-4 MB
- JPEG 50%: 500 KB - 1 MB

Thank You!

Questions?

"JPEG: Making the internet possible since 1992"

Next class: Video compression and how YouTube works!