Lesson Plan: Human Fax Machine

Year Group: 9 | **Duration:** 50 minutes | **Topic:** Binary Encoding & Run-Length Encoding (RLE) Compression

1. Overview

Core Concept: Binary encoding of images (pixel grids) and Run-Length Encoding (RLE) compression — representing data more efficiently by describing runs of identical values.

Learning Objectives:

- Encode an 8×8 binary image as a data stream (row by row, 1=filled, 0=empty)
- Decode a binary stream back into an image (as a "receiver")
- Apply RLE to compress a binary sequence
- Calculate compression ratio and explain why some images compress better than others

Key Vocabulary:

Term	Definition
Binary	A number system using only 0 and 1
Pixel	The smallest unit of a digital image
Encode	Convert data into a specific format for transmission or storage
Decode	Convert encoded data back into its original form
Run-Length Encoding (RLE)	A compression method that replaces consecutive repeated values with a count and value
Compression ratio	Original size ÷ compressed size — how much smaller the compressed version is
Lossless	Compression that allows perfect reconstruction of the original data

2. Before the Lesson

Print:

- ☐ `resource-pixel-images.md` — 1 set per group (contains 4 images, each with a filled and blank version)
- ☐ `worksheet-compression.md` — 1 per student

Room Setup: Pairs. One person ("transmitter") has the filled image. The other ("receiver") has the blank grid.

3. Timed Lesson Flow

0–5 min — Hook: How Does a Photo Travel?

1. *"How does a photo travel across the internet? How did fax machines work in the 1980s?"*
2. It has to become numbers first. Everything digital is ultimately 0s and 1s.
3. Show the concept: take a tiny 4×4 smiley face image. If ■=1 and □=0, how would you write it as numbers?
4. *"That's the idea we'll explore today."*

5–12 min — Introduce 8×8 Encoding

1. Row by row, left to right: ■=1, □=0.
2. Each row produces 8 bits. Full image = 8 rows × 8 bits = 64 bits.
3. Work through Row 1 of Image 1 together.

12–25 min — Human Fax Machine

1. Transmitter looks at the filled pixel image (Image 1 or 2).
2. Receiver has the blank 8×8 grid.
3. Transmitter reads out bits row by row: "Row 1: 0,0,0,0,0,0,0,0"
4. Receiver fills in their grid.
5. At the end: compare. Any errors? Note which rows had transmission errors.
6. *"This is exactly how fax machines work — and how image files are stored."*

25–30 min — Introduce RLE

1. *"Instead of listing every single bit, we can describe RUNS of identical bits."*
2. Example: `0 0 0 1 1 1 1 0 0` → `(3,0)(4,1)(2,0)` — 3 zeros, 4 ones, 2 zeros.
3. Original: 9 values. RLE: 6 numbers. Ratio = 9/6 = 1.5:1.

30–40 min — Worksheet: Apply RLE

Students apply RLE to the Checkerboard and Solid Block images. Calculate and compare compression ratios.

40–47 min — Discussion

- Which image compressed better? Why?
- *"What type of image compresses well with RLE? What type compresses poorly?"*
- Real-world connection: JPEG/PNG/GIF all use compression. RLE is used in fax, BMP. Lossless vs lossy.

47–50 min — Key Takeaway

---

4. Teacher Facilitation Notes

What to look for:

- Transmission errors in the fax activity — celebrate these! They motivate error-detection discussion.

- Students who count individual bits instead of runs for RLE — prompt: *"How many in a ROW before the value changes?"*
- Students confused about compression ratio direction — clarify: ratio > 1 means compression worked.

Common misconceptions:

- RLE always makes files smaller — NO. The checkerboard example shows it can make files LARGER.
- All compression is lossy — RLE is lossless; you can reconstruct the original perfectly.

---

5. Extension Tasks

1. Design a simple error-detection scheme: the transmitter adds a "checksum" bit to each row. How could the receiver detect an error?
2. What would happen if you applied RLE twice to the same data? Would it compress further?
3. Research: what is Huffman coding? How is it different from RLE?

---

6. Key Takeaway

> **All digital images are binary numbers. Compression removes redundancy — repeated patterns compress well, random/varied patterns don't. RLE is one of the simplest compression algorithms.**