

Method

All experiments use 352×288 inputs as required. I quantize either in RGB ($C=1$) or YUV ($C=2$), then always convert back to RGB and evaluate quality with the assignment metric $\text{AbsErr} = \sum |\text{orig} - \text{out}|$ over all pixels and channels. For each $N \in \{4, 6, 8\}$, I enumerate every ordered split (Q_1, Q_2, Q_3) with $Q_1 + Q_2 + Q_3 = N$ under both $M=1$ (uniform) and $M=2$ (smart), generate the output, and record AbsErr . Per spec, I show all 12 images for $N=4$ (3 splits \times 4 (C,M) pairs) and provide numbers only for $N=6/8$. As a sanity check, an $8/8/8$ round-trip $\text{RGB} \leftrightarrow \text{YUV} \leftrightarrow \text{RGB}$ reconstructs the input ($\text{AbsErr} \approx 0$).

Discussion of Results

Case 1: $C=1, M=1$ (RGB + Uniform)

Uniform bins don't adapt to the image, so starving any channel hurts. In my 352×288 runs, the tight $N=4$ budget is essentially a three-way trade; the best RGB-uniform split is $(2, 1, 1)$ by a small margin. As the budget grows, the trend toward balance shows up clearly: at $N=6$ the winner is $(2, 2, 2)$, and at $N=8$ the lowest error is $(3, 2, 3)$ (balanced or near-balanced). In short, once you have more bits to spread, keeping the channels roughly even minimizes AbsErr .

Case 2: $C=1, M=2$ (RGB + Smart)

Smart, data-adaptive quantization is the clear winner in RGB at this resolution. In my 352×288 runs, the best $N=4$ split is $(1, 2, 1)$ with the lowest $\text{AbsErr} = 3,306,703$. As the budget increases, balanced allocations dominate: $(2, 2, 2)$ wins at $N=6$ with $\text{AbsErr} = 1,723,361$, and $(2, 3, 3)$ wins at $N=8$ with $\text{AbsErr} = 1,048,004$. In short, adapting the bins to the data slashes error versus uniform, and keeping R/G/B roughly balanced (with a slight tilt toward G/B at higher N) gives the best AbsErr .

Case 3: $C=2, M=1$ (YUV + Uniform)

Uniform binning in YUV is the weakest overall, but the pattern is clear in my 352×288 runs. With the tight $N=4$ budget, the best split is $(2, 1, 1)$ (more bits on Y). As the budget grows, balanced splits start to win: at $N=6$ the lowest error is $(2, 2, 2)$, and at $N=8$ it's $(3, 2, 3)$ (near-balanced). This matches intuition—uniform chroma bins don't match the clustered U/V distributions, so starving either chroma channel hurts; spreading bits more evenly reduces that penalty, though YUV-uniform still trails the smart methods by a wide margin in AbsErr .

Case 4: $C=2, M=2$ (YUV + Smart)

YUV+smart improves a lot over YUV uniform, but on my 352×288 runs it still trails RGB+smart at the same N . With the very tight $N=4$ budget the best split is $(1, 1, 2)$ (skewing a bit toward V)

with AbsErr = 4,202,330. As soon as the budget opens up, the pattern shifts toward balance: at N=6 the winner is (2,2,2) with AbsErr = 2,542,065, and at N=8 the best is (3,2,3) at 1,318,064. So while YUV+smart benefits from allocating more to luminance in general, the lowest errors here arrive once U/V aren't starved and the split is balanced or near-balanced—still not enough to surpass the RGB+smart winners at each N.

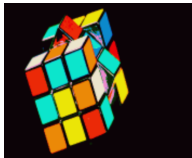
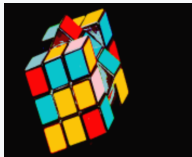
Overall Conclusion






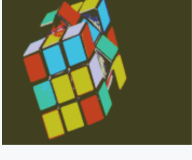
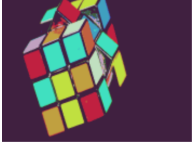
Across all three budgets (N=4,6,8) on the 352×288 image, smart quantization consistently beats uniform, and the overall winners are RGB + smart (C=1, M=2) with balanced or near-balanced splits. Concretely, the lowest-error rows are (1,2,1) at N=4, (2,2,2) at N=6, and (2,3,3) at N=8—each minimizing AbsErr more than any YUV configuration. YUV + smart improves a lot over YUV uniform, but even its best rows (which trend toward balanced splits once the budget allows) still trail the RGB + smart winners by AbsErr. The simple rule that held up in my results: use smart quantization; in RGB keep bits roughly balanced; if you work in YUV, don't starve U or V.




What I'm submitting

- N=4 table with images and N=6/N=8 tables with error numbers, all directly from analysis.csv (sorted by AbsErr ascending per N).
- This narrative section referencing the lowest-error tuples above.
- Reproducibility: the CSV was generated by my Analysis target; N=4 PNGs are named as shown in the table.
- [Link to spreadsheet with all analysis and images](#)
- [Link to Google doc of this analysis which contains image embeds](#)

N = 4

N	<C,M,Q1,Q2,Q3>	Output	Error
4	<1,2,1,2,1>		3306703
4	<1,2,2,1,1>		3380230

4	$\langle 1,2,1,1,2 \rangle$		3404664
4	$\langle 2,2,1,1,2 \rangle$		4202330
4	$\langle 2,2,2,1,1 \rangle$		6275459
4	$\langle 2,2,1,2,1 \rangle$		6753180
4	$\langle 1,1,2,1,1 \rangle$		13737203
4	$\langle 1,1,1,1,2 \rangle$		13846799
4	$\langle 1,1,1,2,1 \rangle$		13927589

4	$\langle 2,1,2,1,1 \rangle$		16233553
4	$\langle 2,1,1,2,1 \rangle$		19014671
4	$\langle 2,1,1,1,2 \rangle$		19745007

N=6

N	$\langle C,M,Q1,Q2,Q3 \rangle$	Error
6	$\langle 1,2,2,2,2 \rangle$	1723361
6	$\langle 1,2,2,3,1 \rangle$	2152427
6	$\langle 1,2,1,3,2 \rangle$	2176861
6	$\langle 1,2,1,2,3 \rangle$	2202280
6	$\langle 1,2,3,2,1 \rangle$	2248828
6	$\langle 1,2,2,1,3 \rangle$	2275807
6	$\langle 1,2,3,1,2 \rangle$	2346789
6	$\langle 2,2,2,2,2 \rangle$	2542065
6	$\langle 2,2,3,1,2 \rangle$	2770705
6	$\langle 1,2,1,4,1 \rangle$	2851275
6	$\langle 1,2,1,1,4 \rangle$	3007407
6	$\langle 1,2,4,1,1 \rangle$	3015091
6	$\langle 2,2,2,1,3 \rangle$	3054208
6	$\langle 2,2,1,2,3 \rangle$	3546407
6	$\langle 2,2,1,3,2 \rangle$	3728643
6	$\langle 2,2,1,1,4 \rangle$	3987587

6	<2,2,3,2,1>	4972420
6	<2,2,2,3,1>	5573644
6	<2,2,4,1,1>	5810447
6	<2,2,1,4,1>	6733654
6	<1,1,2,2,2>	7890745
6	<2,1,2,2,2>	8973445
6	<1,1,3,1,2>	9291649
6	<1,1,2,1,3>	9357907
6	<1,1,3,2,1>	9372439
6	<1,1,2,3,1>	9458647
6	<1,1,1,2,3>	9548293
6	<1,1,1,3,2>	9568243
6	<2,1,3,1,2>	10036539
6	<2,1,3,2,1>	10218836
6	<2,1,2,1,3>	11060419
6	<1,1,4,1,1>	11601643
6	<1,1,1,1,4>	11730021
6	<1,1,1,4,1>	11829199
6	<2,1,2,3,1>	12755721
6	<2,1,4,1,1>	12874113
6	<2,1,1,2,3>	15957858
6	<2,1,1,3,2>	17148474
6	<2,1,1,1,4>	18540844
6	<2,1,1,4,1>	19214042

N=8

N	<C,M,Q1,Q2,Q3>	Error
8	<1,2,2,3,3>	1048004
8	<1,2,3,3,2>	1118986
8	<1,2,3,2,3>	1144405
8	<1,2,2,4,2>	1267933
8	<2,2,3,2,3>	1318064

8	<1,2,2,2,4>	1326104
8	<1,2,4,2,2>	1358222
8	<2,2,4,2,2>	1535686
8	<2,2,3,3,2>	1621811
8	<1,2,1,4,3>	1746852
8	<1,2,1,3,4>	1779604
8	<1,2,4,3,1>	1787288
8	<1,2,3,4,1>	1793400
8	<1,2,4,1,3>	1910668
8	<1,2,3,1,4>	1949532
8	<1,2,2,5,1>	1954366
8	<1,2,1,5,2>	1978800
8	<1,2,5,2,1>	2060433
8	<1,2,1,2,5>	2063191
8	<1,2,2,1,5>	2136718
8	<2,2,2,2,4>	2142779
8	<1,2,5,1,2>	2158394
8	<2,2,2,3,3>	2177222
8	<2,2,4,1,3>	2302858
8	<2,2,3,1,4>	2307437
8	<2,2,2,4,2>	2481419
8	<2,2,5,1,2>	2595550
8	<1,2,1,6,1>	2706083
8	<1,2,1,1,6>	2875109
8	<1,2,6,1,1>	2901044
8	<2,2,2,1,5>	2967421
8	<2,2,1,2,5>	3501503
8	<2,2,1,3,4>	3519880
8	<2,2,1,4,3>	3536108
8	<2,2,1,5,2>	3698295
8	<2,2,1,1,6>	3947865
8	<2,1,3,2,3>	4391327
8	<2,2,4,3,1>	4803941

8	<2,2,5,2,1>	4843974
8	<2,2,3,4,1>	4879087
8	<1,1,3,2,3>	4993143
8	<1,1,3,3,2>	5013093
8	<2,1,4,2,2>	5023253
8	<1,1,2,3,3>	5079351
8	<2,2,2,5,1>	5514033
8	<2,2,6,1,1>	5713151
8	<1,1,4,2,2>	5755185
8	<1,1,2,2,4>	5773967
8	<1,1,2,4,2>	5792355
8	<2,1,3,3,2>	6598811
8	<2,2,1,6,1>	6705228
8	<2,1,4,1,3>	6910694
8	<1,1,3,1,4>	7174871
8	<2,1,3,1,4>	7186082
8	<1,1,4,1,3>	7222347
8	<1,1,3,4,1>	7274049
8	<1,1,4,3,1>	7323087
8	<1,1,1,4,3>	7449903
8	<1,1,1,3,4>	7451465
8	<2,1,2,2,4>	7513444
8	<2,1,2,3,3>	8022169
8	<2,1,5,1,2>	8329245
8	<1,1,2,1,5>	8338749
8	<1,1,2,5,1>	8403739
8	<1,1,5,1,2>	8477229
8	<1,1,1,5,2>	8513335
8	<2,1,5,2,1>	8525464
8	<1,1,1,2,5>	8529135
8	<1,1,5,2,1>	8558019
8	<2,1,2,4,2>	8862125
8	<2,1,4,3,1>	9639872

8	<2,1,3,4,1>	10105439
8	<2,1,2,1,5>	10387529
8	<1,1,1,1,6>	11237411
8	<1,1,6,1,1>	11353565
8	<1,1,1,6,1>	11362595
8	<2,1,6,1,1>	12155116
8	<2,1,2,5,1>	12403487
8	<2,1,1,2,5>	15381150
8	<2,1,1,3,4>	15942894
8	<2,1,1,4,3>	16161680
8	<2,1,1,5,2>	16902761
8	<2,1,1,1,6>	18332109
8	<2,1,1,6,1>	19052594