SM LAB_5 Sprawozdanie
Lewicki Maciej

1. Do zaimplementowania:
    1. Algorytm realizujący nasz uproszczony algorytm JPEG dający wybór pomiędzy 4 kombinacjami poniższych kombinacji: (0.7 pkt)
        1. Redukcję chrominancji (wybór pomiędzy 4:4:4 a 4:2:2)
        2. Wybór tablicy kwantyzującej lub zastąpienie jej tablicą jedynek

1.1.1
Redukcja chrominancji:

```
def chromaSubsample(_C, mode):
    if mode == "4:4:4":
        return _C

    out = np.empty((_C.shape[0], int(_C.shape[1]/2)))
    if mode == "4:2:2":
        for row in range(0, _C.shape[0]):
            for col in range(0, _C.shape[1], 2):
                out[row][int(col/2)] = _C[row][col]
        return out


def chromaUnsample(_C, mode):
    if mode == "4:4:4":
        return _C

    out = np.empty((_C.shape[0], _C.shape[1]*2))
    if mode == "4:2:2":
        for row in range(0, _C.shape[0]):
            for col in range(0, _C.shape[1]):
                out[row][col*2] = _C[row][col]
                out[row][col*2+1] = _C[row][col]
        return out
```

1.1.2

Kompletny algorytm uproszczonej kompresji jpg:

```
def compress(channel, samplingMode, qTable):
    sampled = chromaSubsample(channel, samplingMode)
    sampled = sampled.astype(int) - 128
    dct = dct2(sampled)

    out = np.zeros(sampled.shape[0]*sampled.shape[1])
    indx = 0
    for row in range(0, dct.shape[0], 8):
        for col in range(0, dct.shape[1], 8):
            zz = dct[row:row+8, col:col+8]
            temp = zigzag(zz)
            out[indx:indx+64] = np.round(temp/qTable.flatten()).astype(int)
            indx += 64
    return out


def decompress(channel, samplingMode, qTable):
    if samplingMode ==  "4:2:2":
        out = np.zeros((int(np.sqrt(channel.shape[0]*2)), int(np.sqrt(channel.shape[0]*2)/2)))
    else:
        out = np.zeros((int(np.sqrt(channel.shape[0])), int(np.sqrt(channel.shape[0]))))

    for idx, i in enumerate(range(0, channel.shape[0], 64)):
        dequantized = channel[i:i+64] * qTable.flatten()
        unzigzaged = zigzag(dequantized)

        x = (idx*8) % out.shape[1]
        y = int((idx*8)/out.shape[1])*8
        out[y:y+8, x:x+8] = unzigzaged

    undcted = idct2(out)+128
    undcted = np.clip(undcted, 0, 255).astype(np.uint8)
    unsampled = chromaUnsample(undcted, samplingMode)
    return unsampled
```

2. Sprawozdanie/raport z działania programu (0,3 pkt):
    1. Wybrać kilka (więcej niż 2) różnych dużych zdjęć i przeanalizować mniejsze ich fragmenty (najlepiej kilka na jednym reprezentująca różne sytuacje obrazu). I porównać ich działanie dla różnych wariantów działania naszego algorytmu. W sumie na każdym wycinku do sprawdzenia na każdym są 4 warianty. Pamiętać, żeby załączane obrazy były czytelne, czyli nie załączać zrzutów ekranów tylko zapisane ploty i starajcie się tak je projektować żeby w PDFie nie uległy one pomniejszaniu, bo kompresja dokłada dodatkowe artefakty.
    2. wycinki rozmiaru 128x128, 256x256

2.

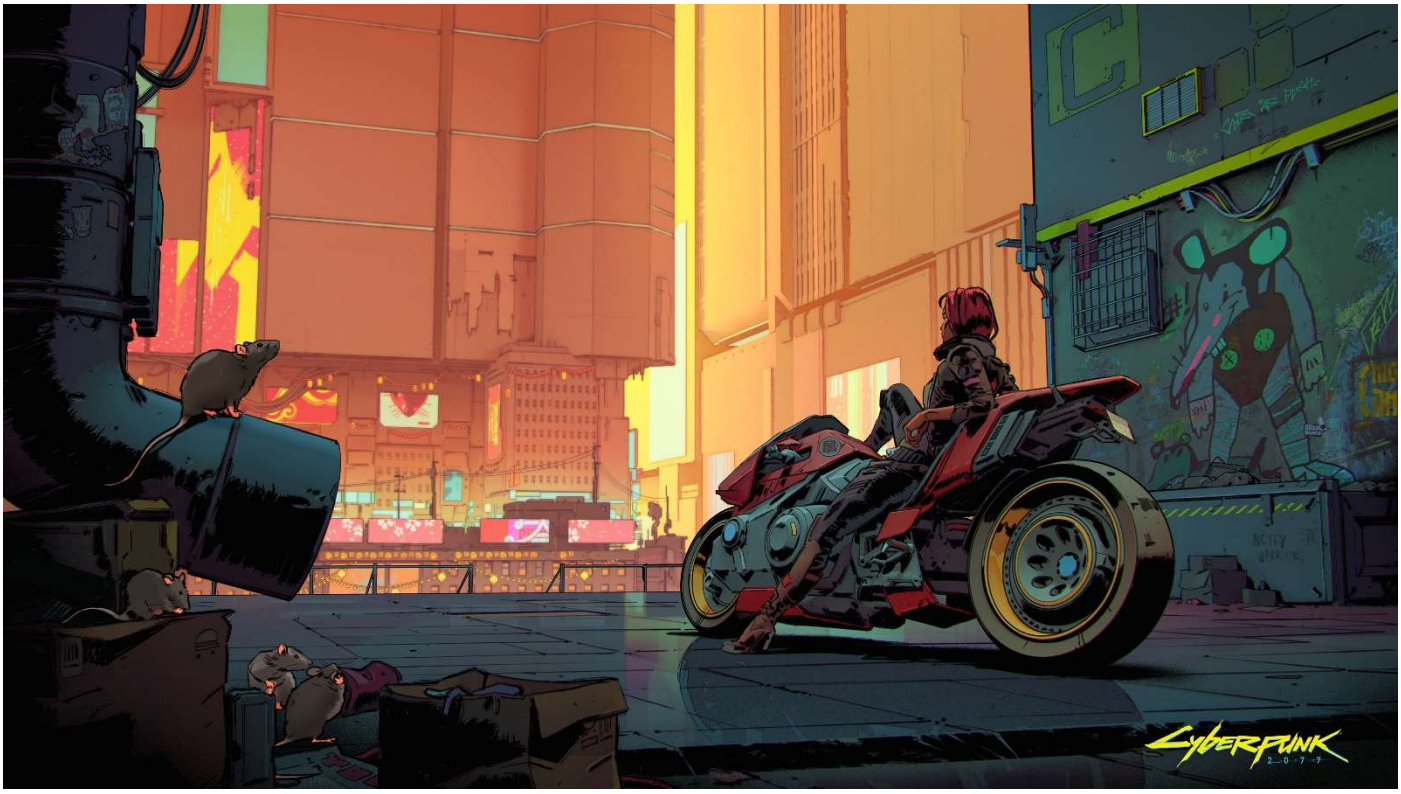
Wyniki:

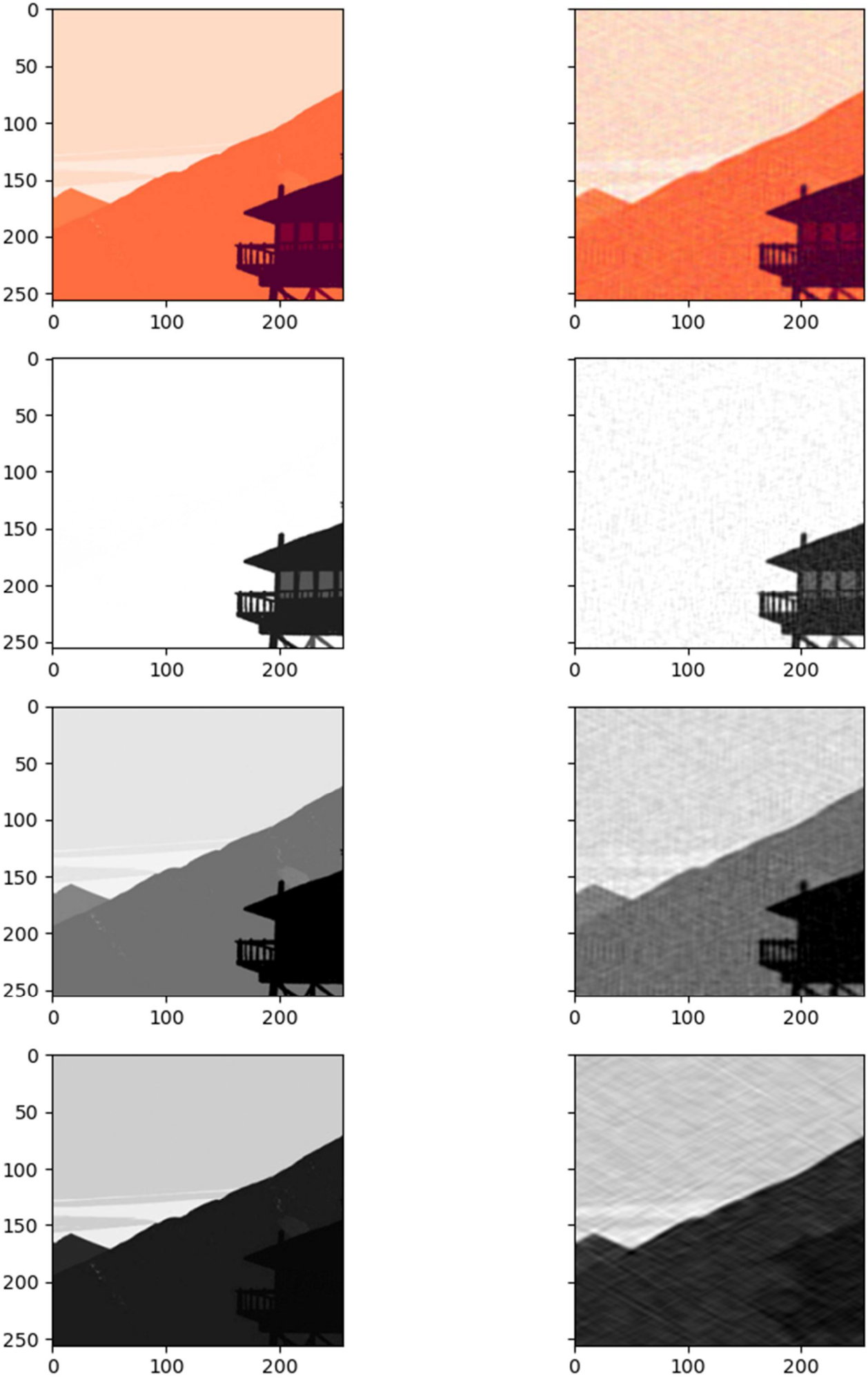
Testowane obrazy:

1.

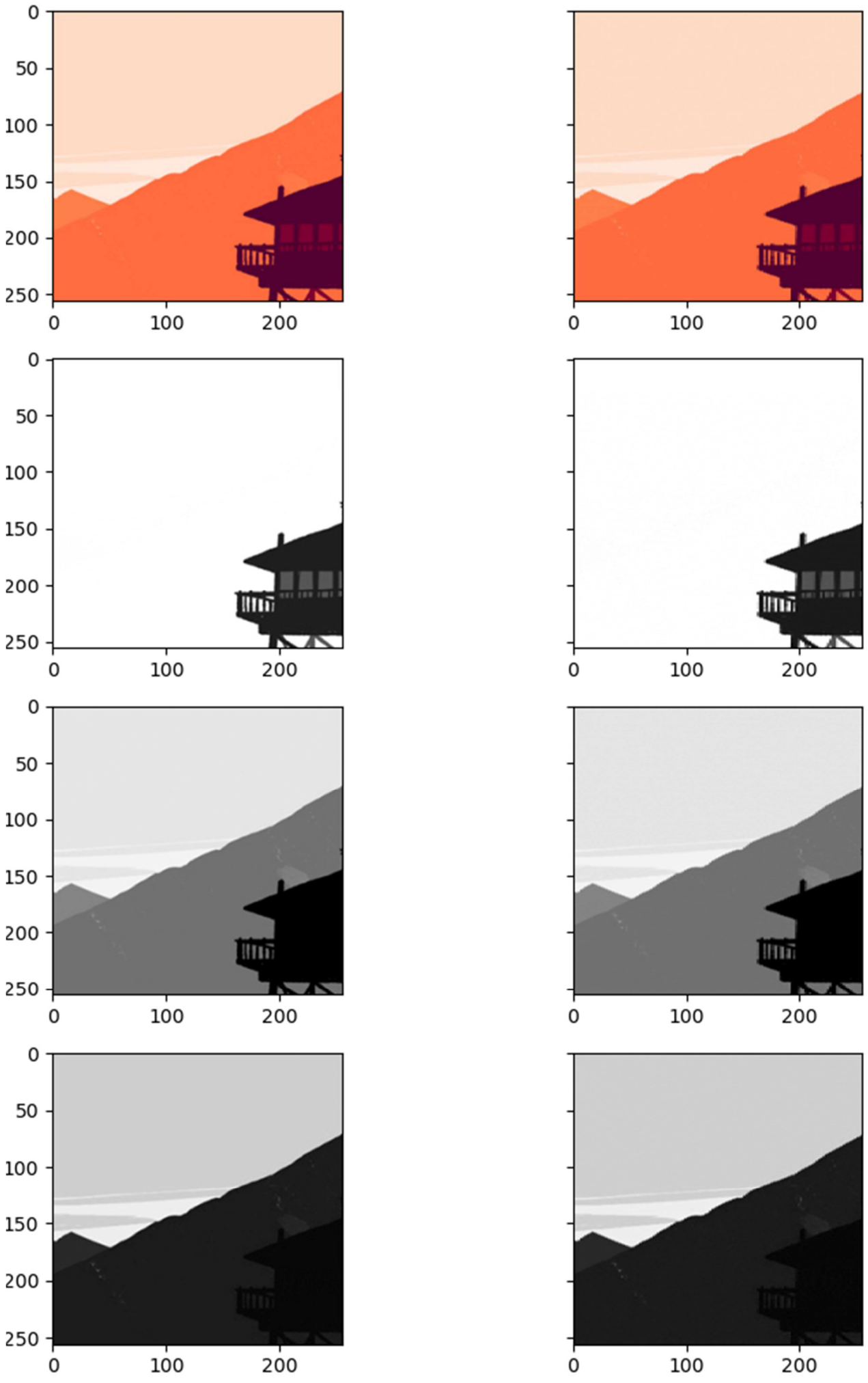
2.

3.

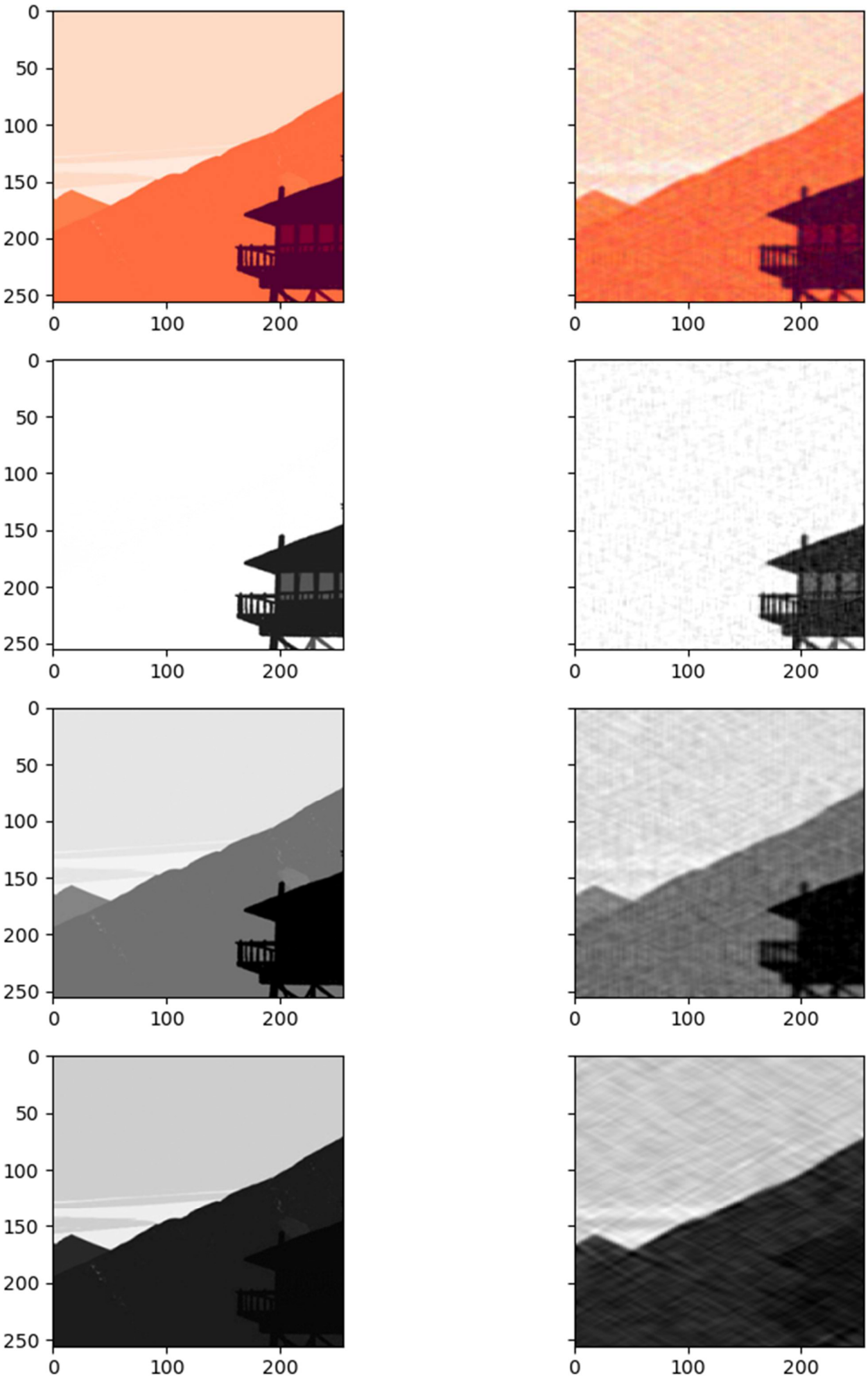




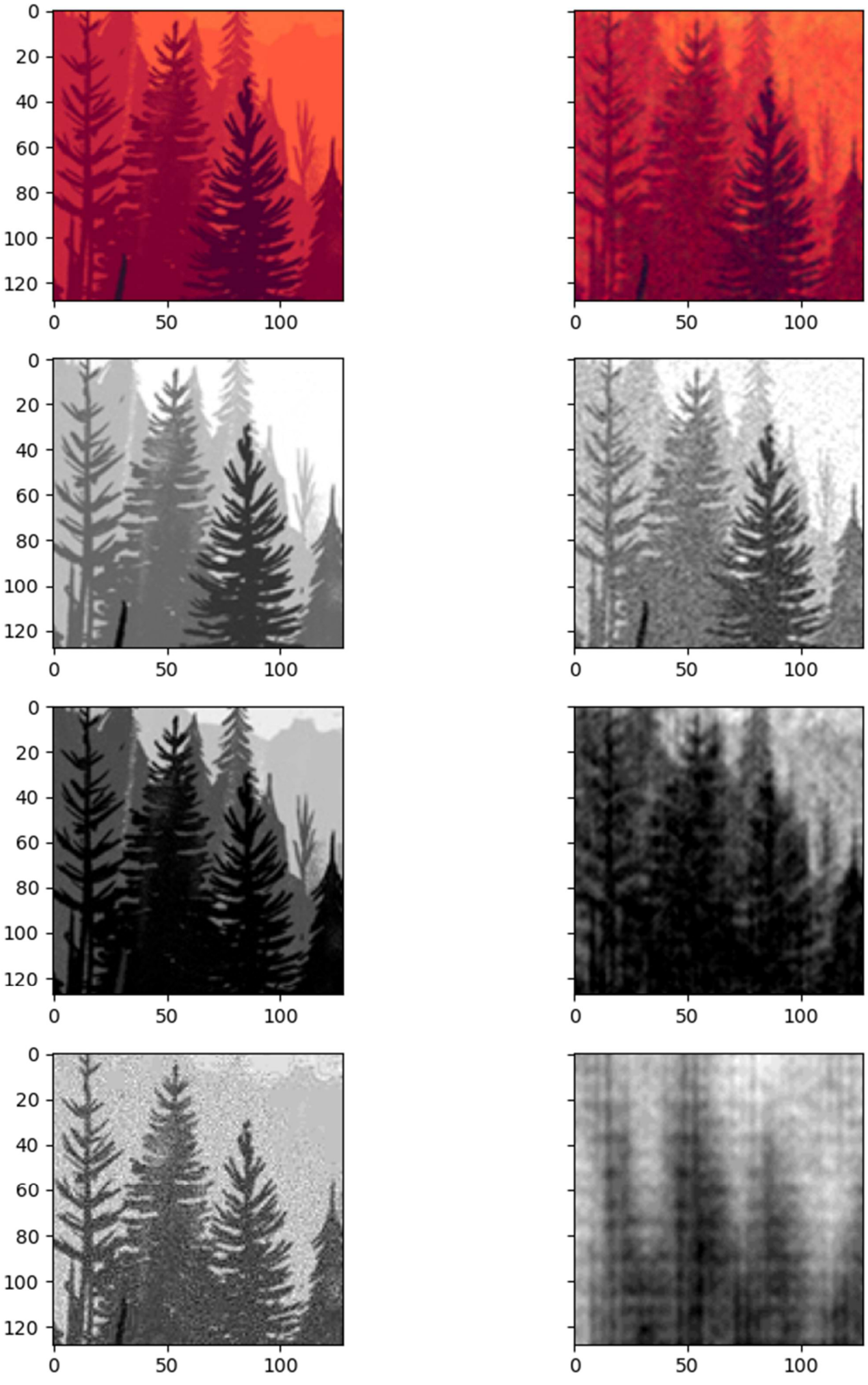
Subsampling 4:4:4, lum, chrom 50%, 256x256 tło + detale



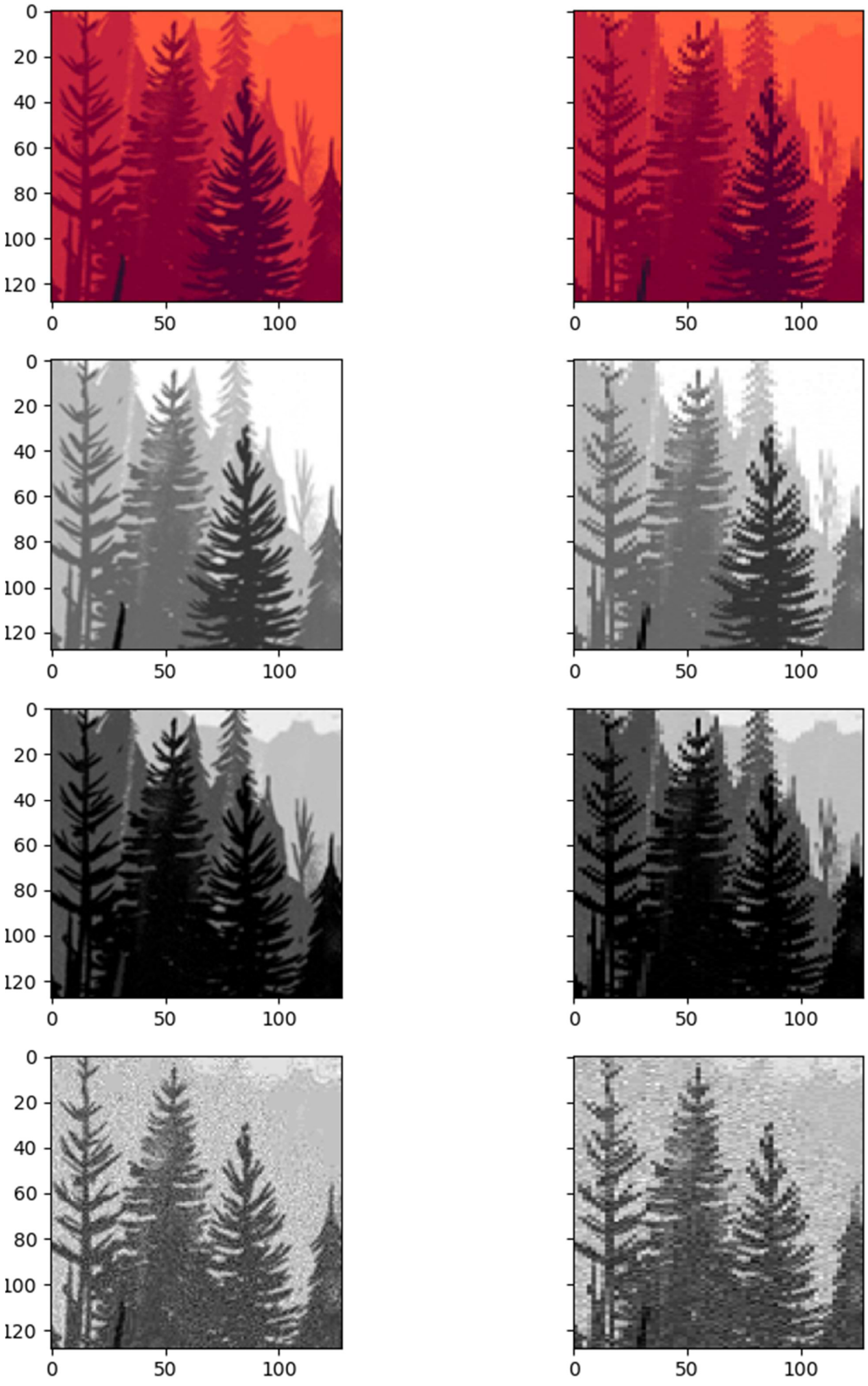
Subsampling 4:2:2, lum, chrom 100%, 256x256 tło + detale



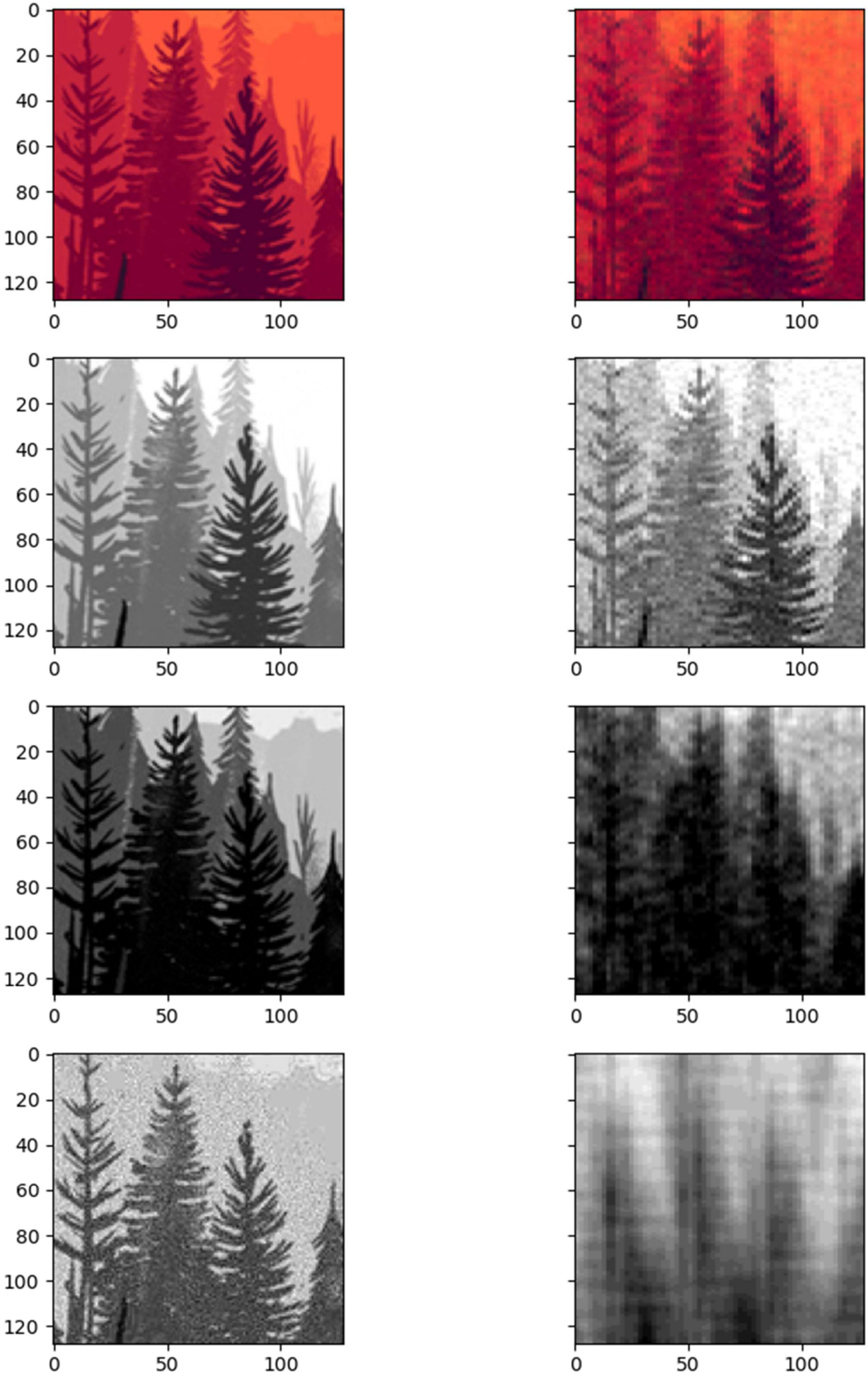
Subsampling 4:2:2, lum, chrom 50%, 256x256 tło + detale



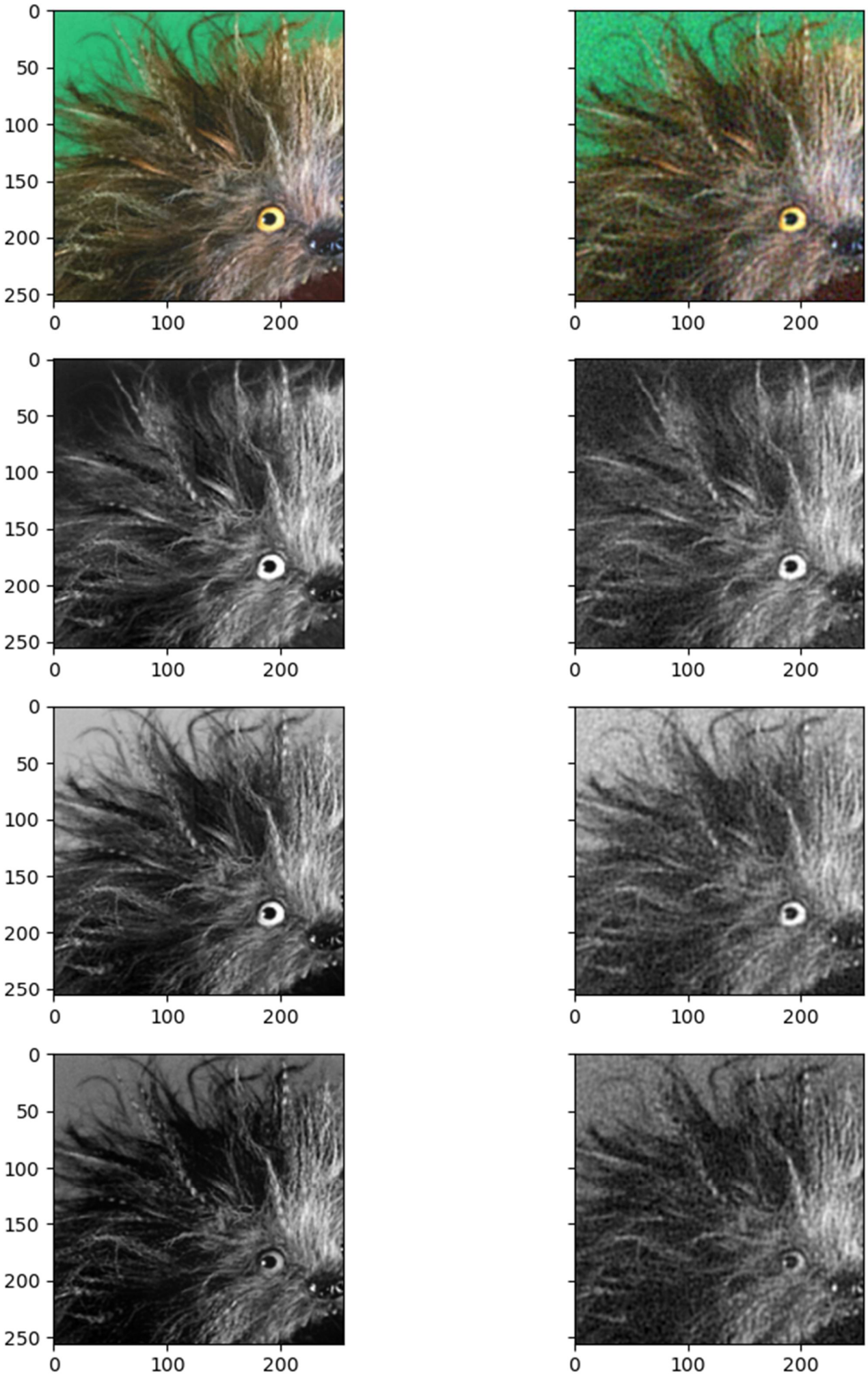
Subsampling 4:4:4, lum, chrom 50%, 128x128 detale



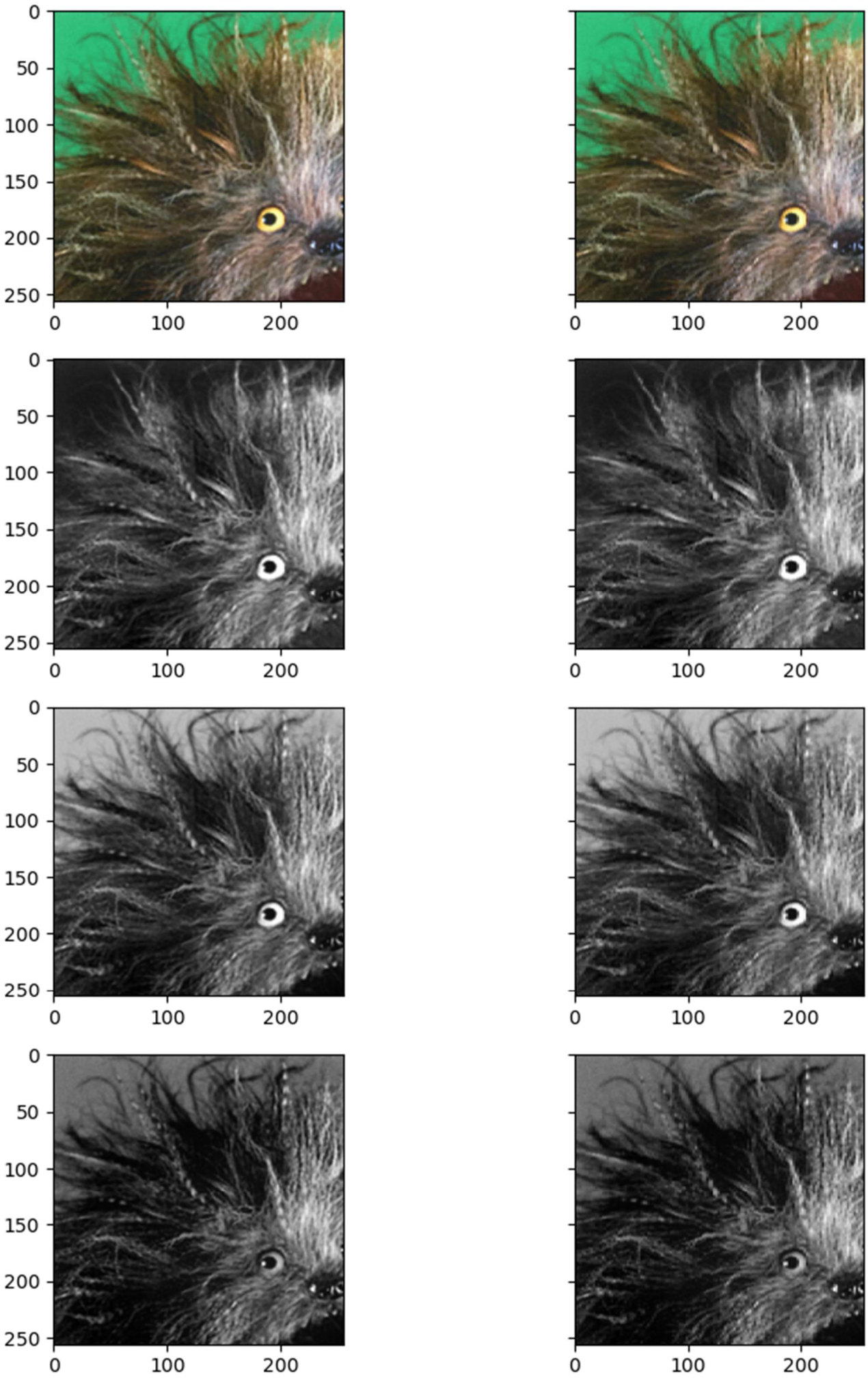
Subsampling 4:2:2, lum, chrom 100%, 128x128 detale



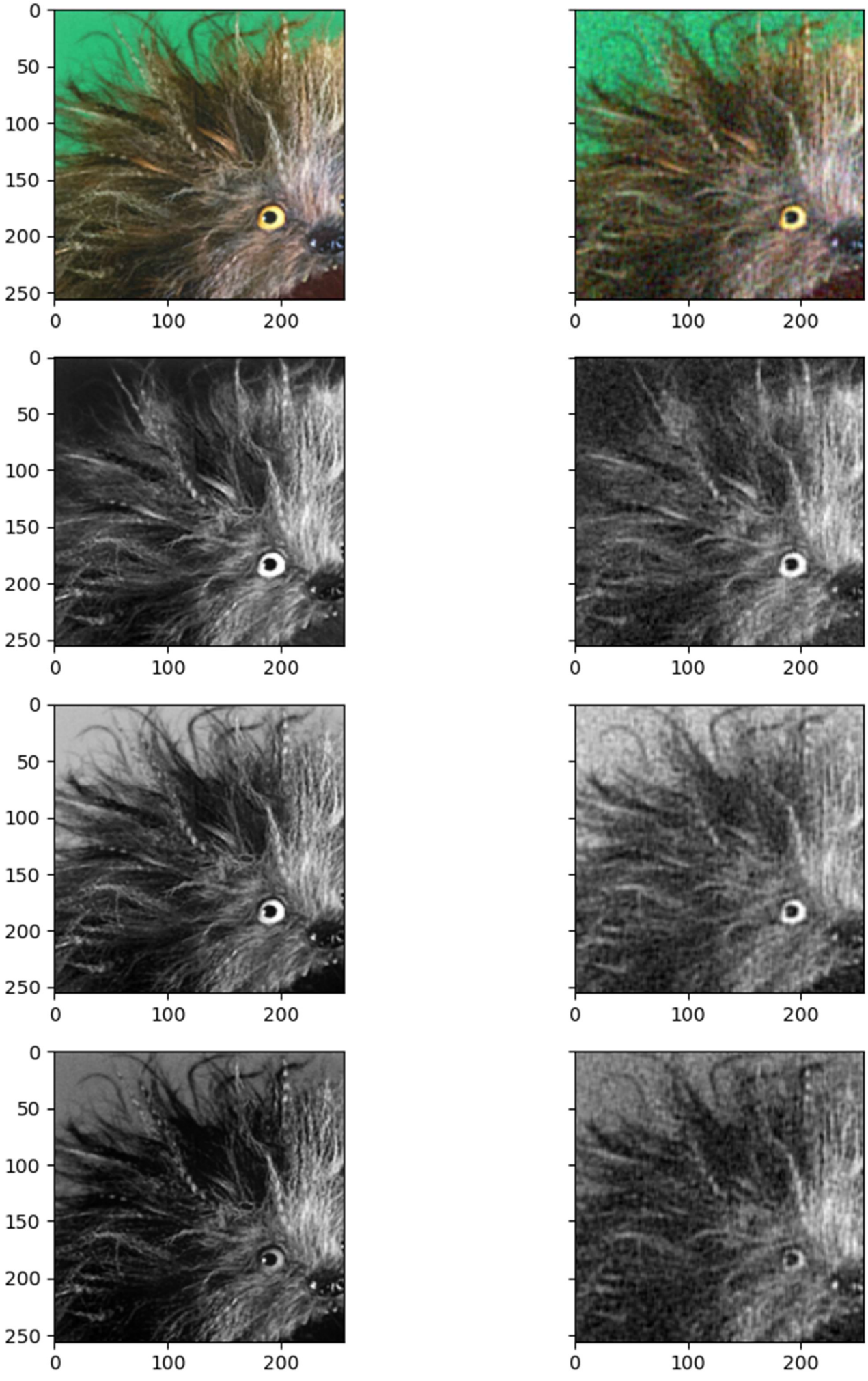
Subsampling 4:2:2, lum, chrom 50%, 128x128 detale



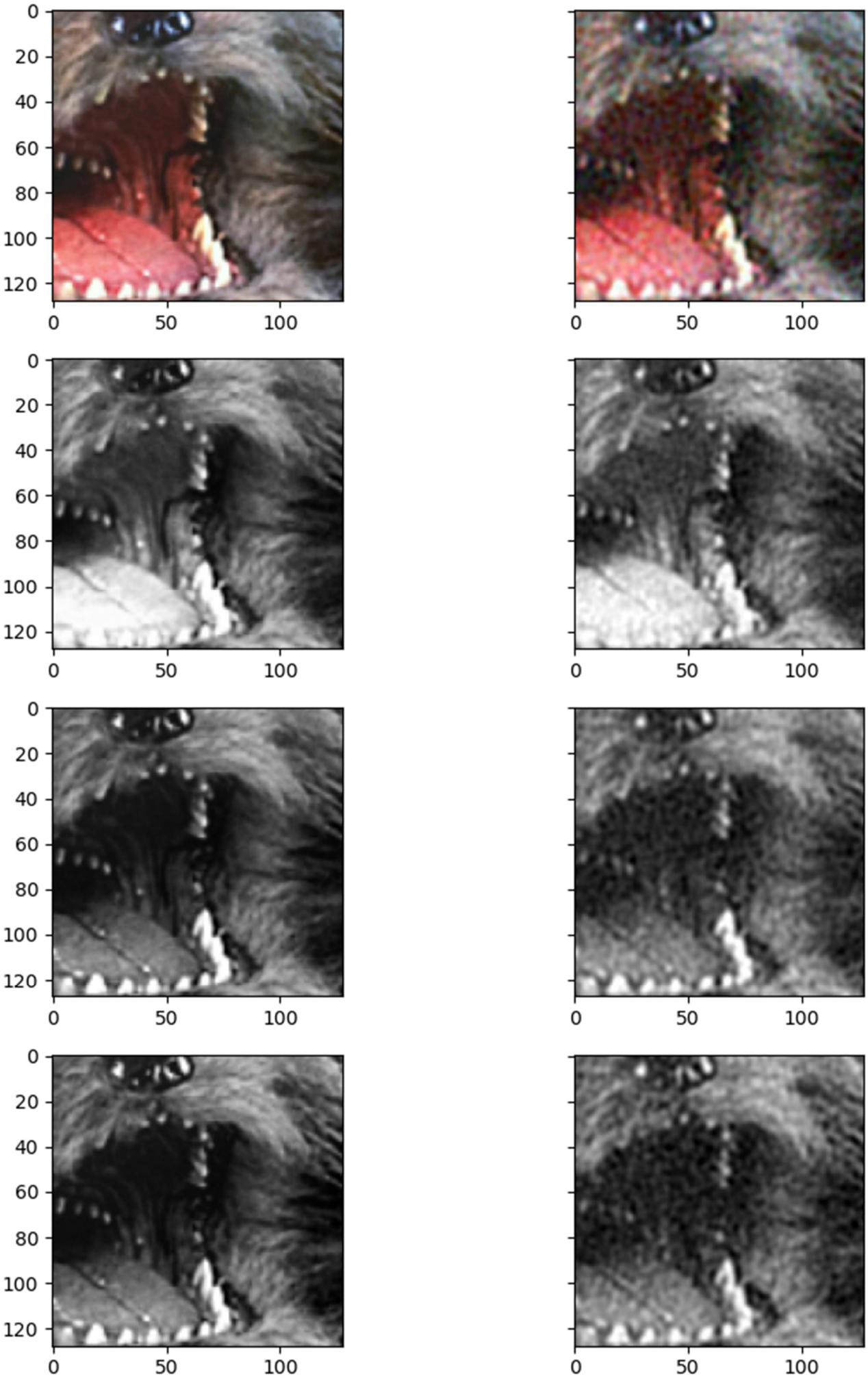
Subsampling 4:4:4, lum, chrom 50%, 256x256



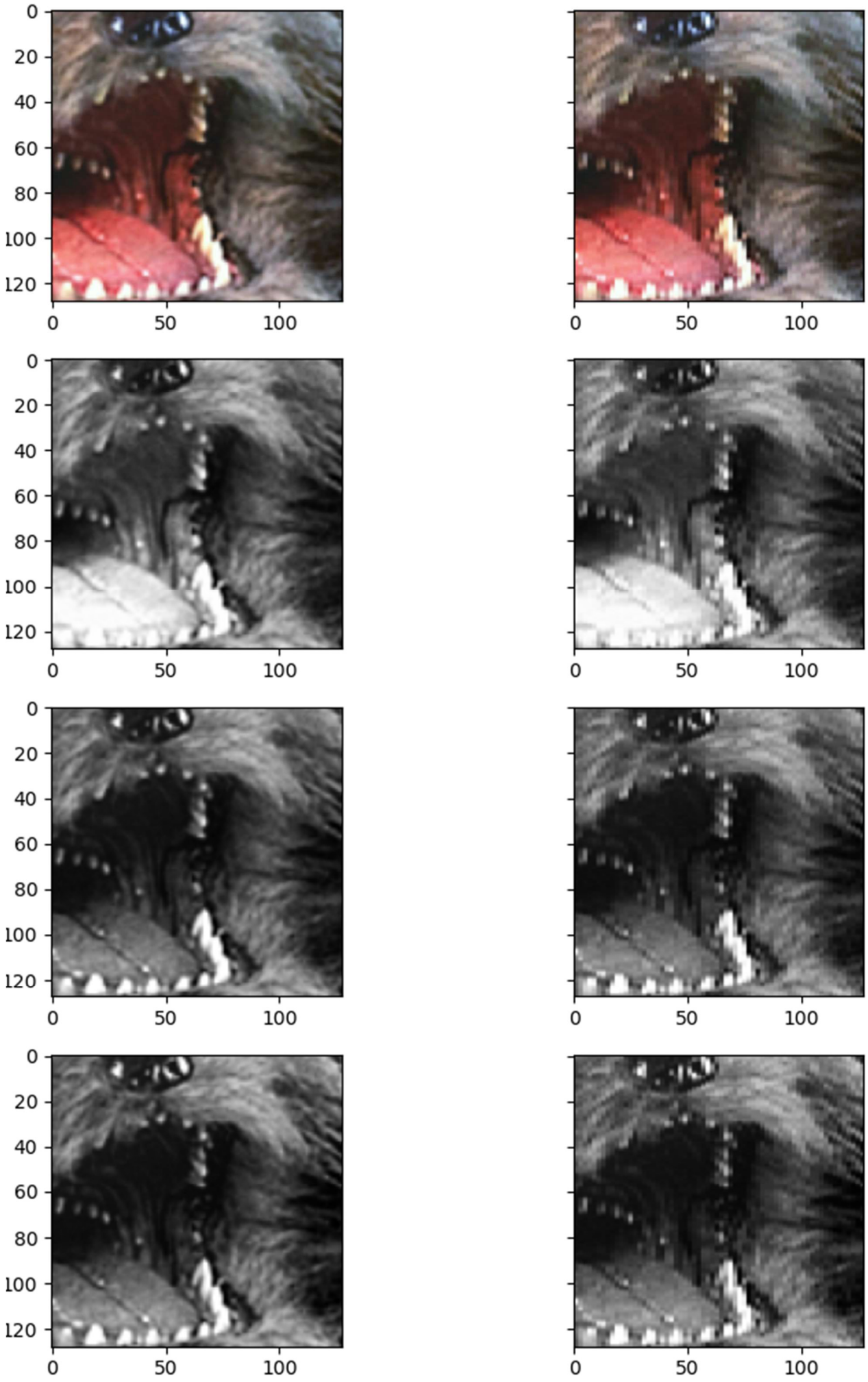
Subsampling 4:2:2, lum, chrom 100%, 256x256



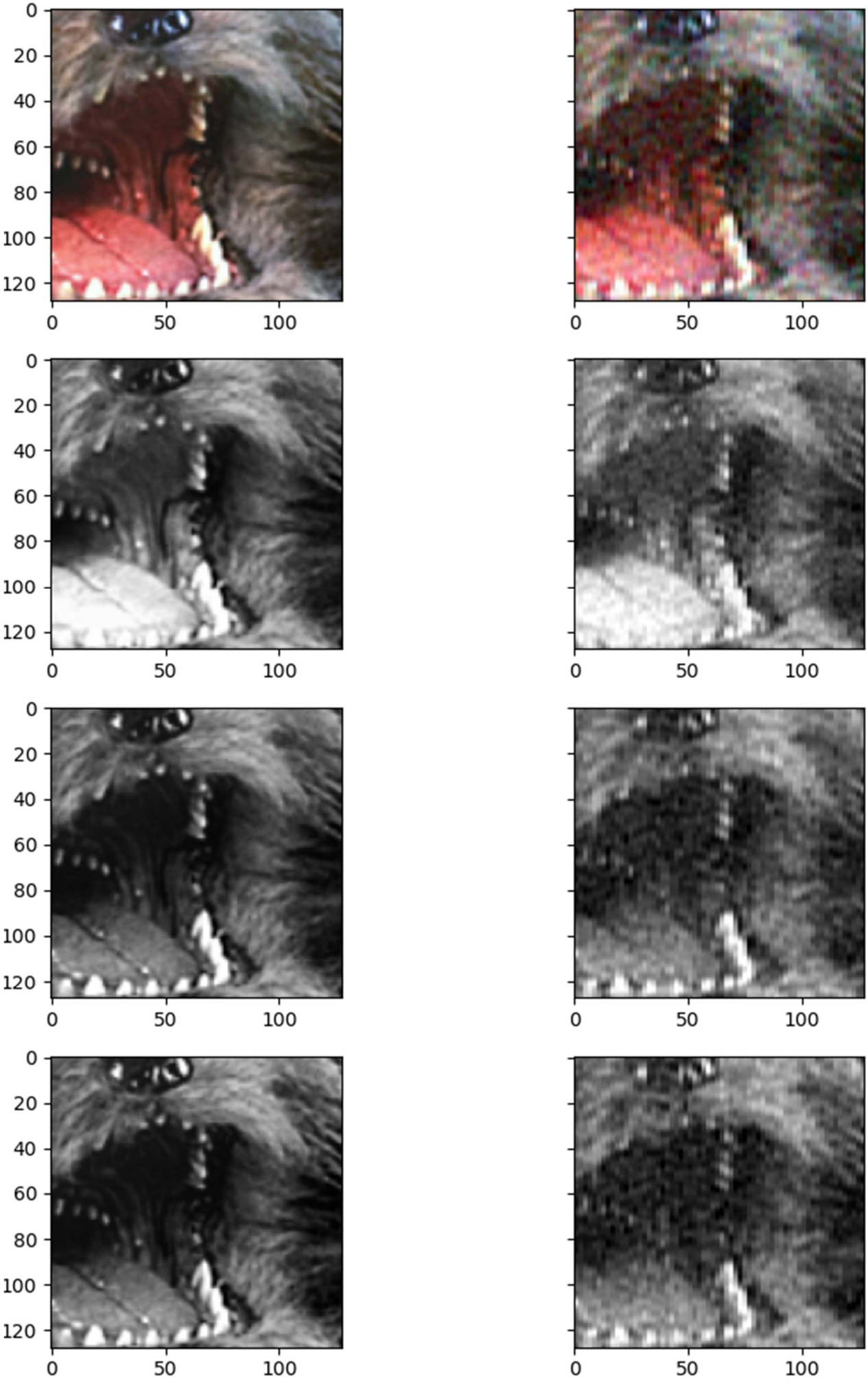
Subsampling 4:2:2, lum, chrom 50%, 256x256



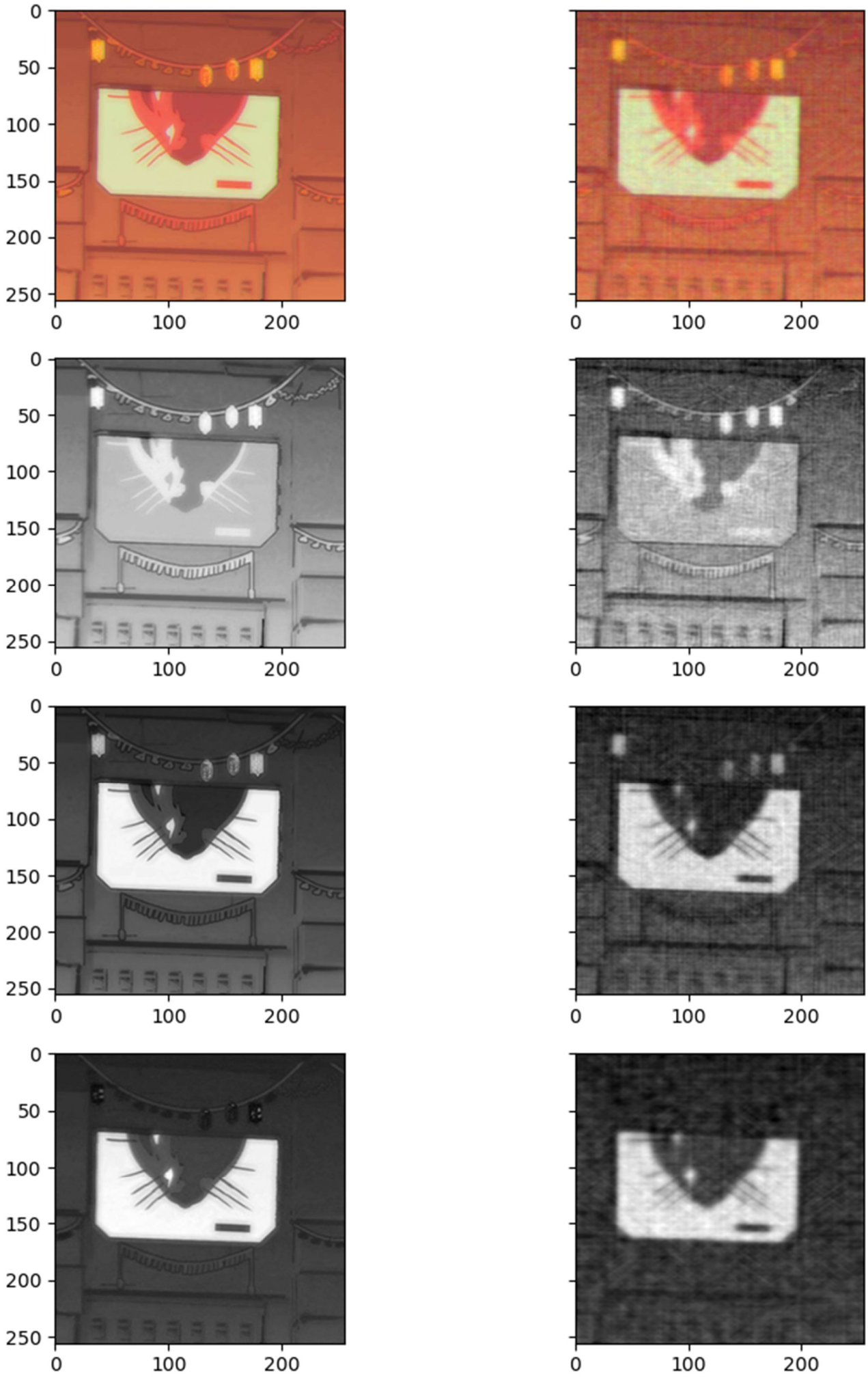
Subsampling 4:4:4, lum, chrom 50%, 128x128



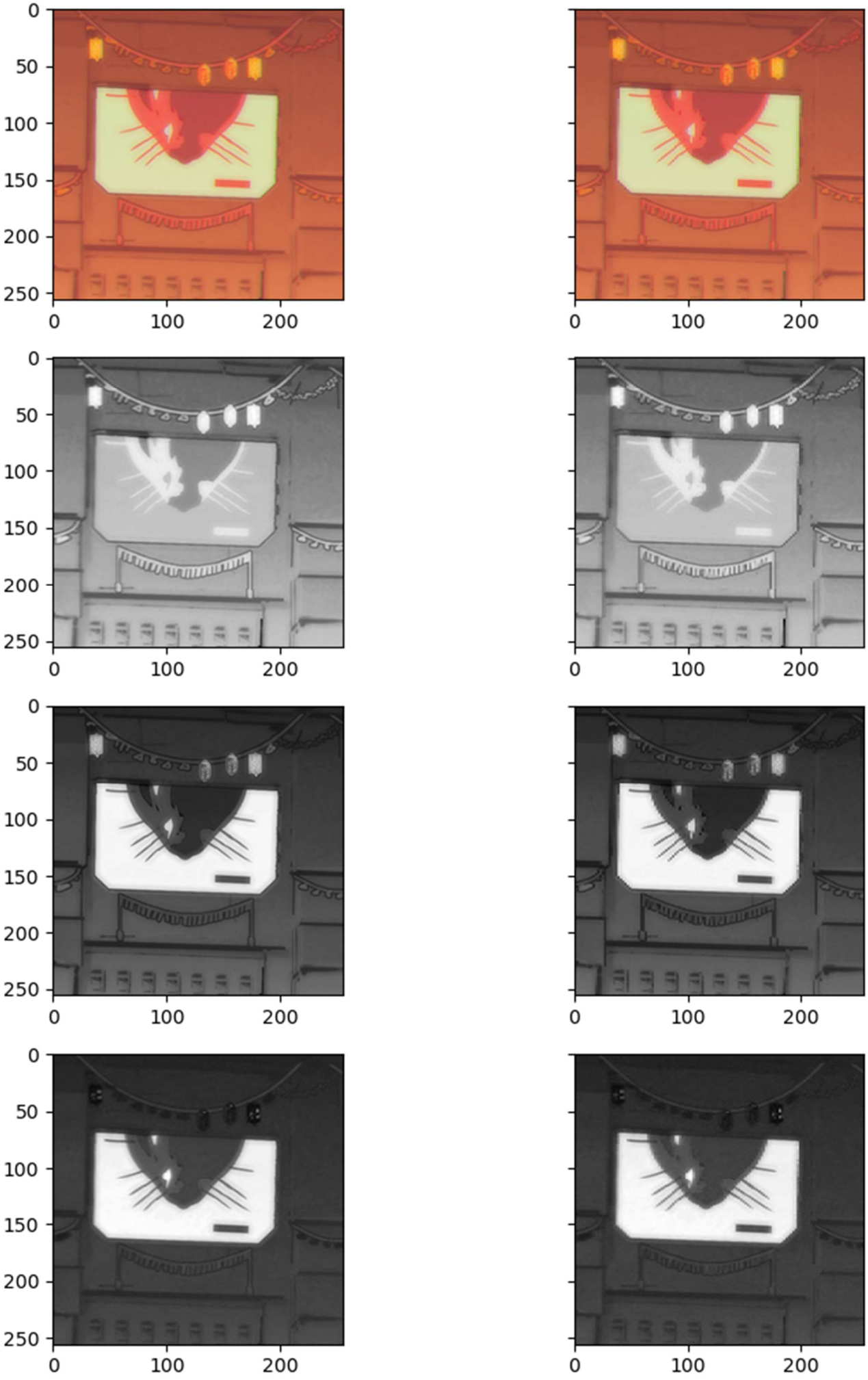
Subsampling 4:2:2, lum, chrom 100%, 128x128



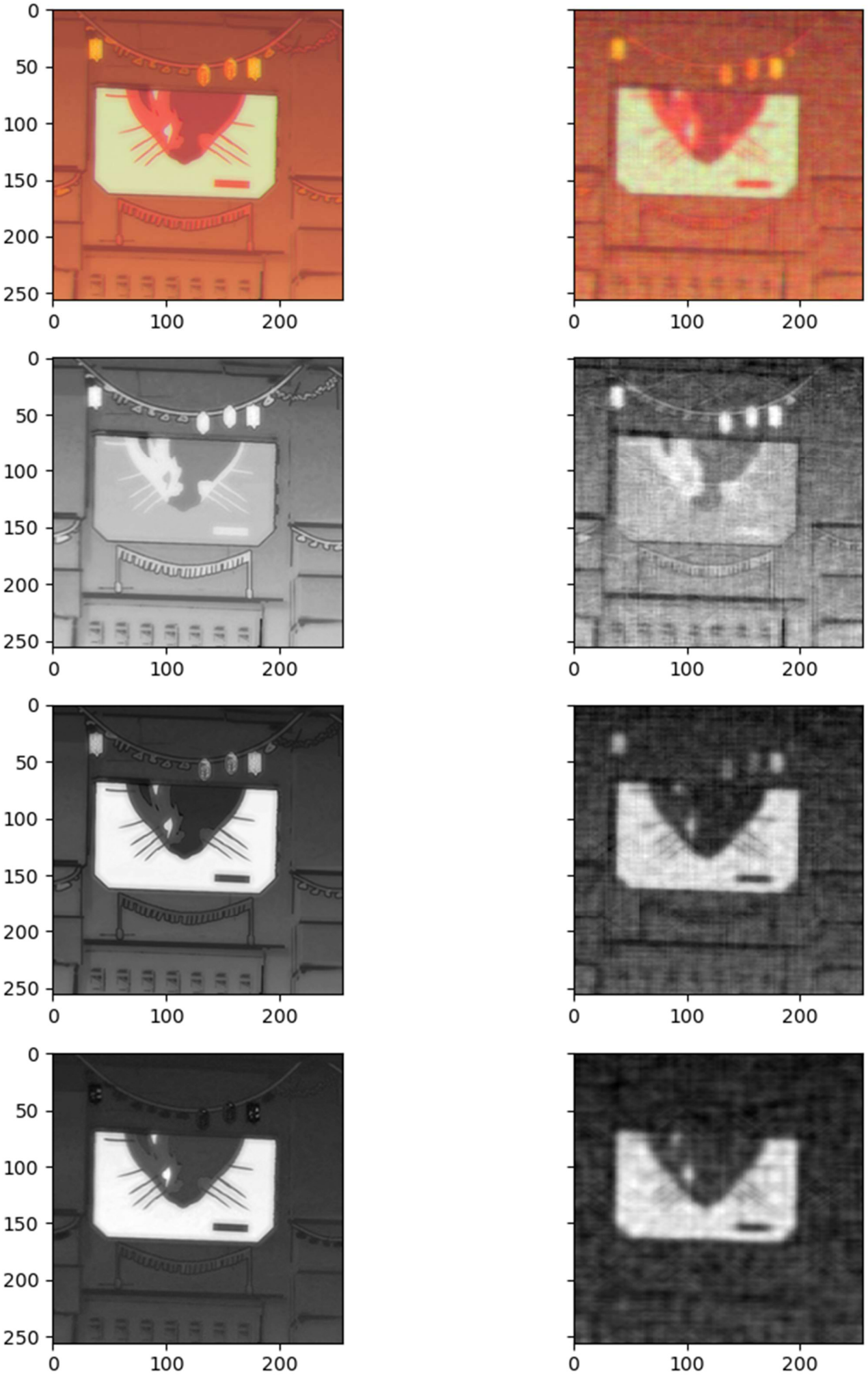
Subsampling 4:2:2, lum, chrom 50%, 128x128



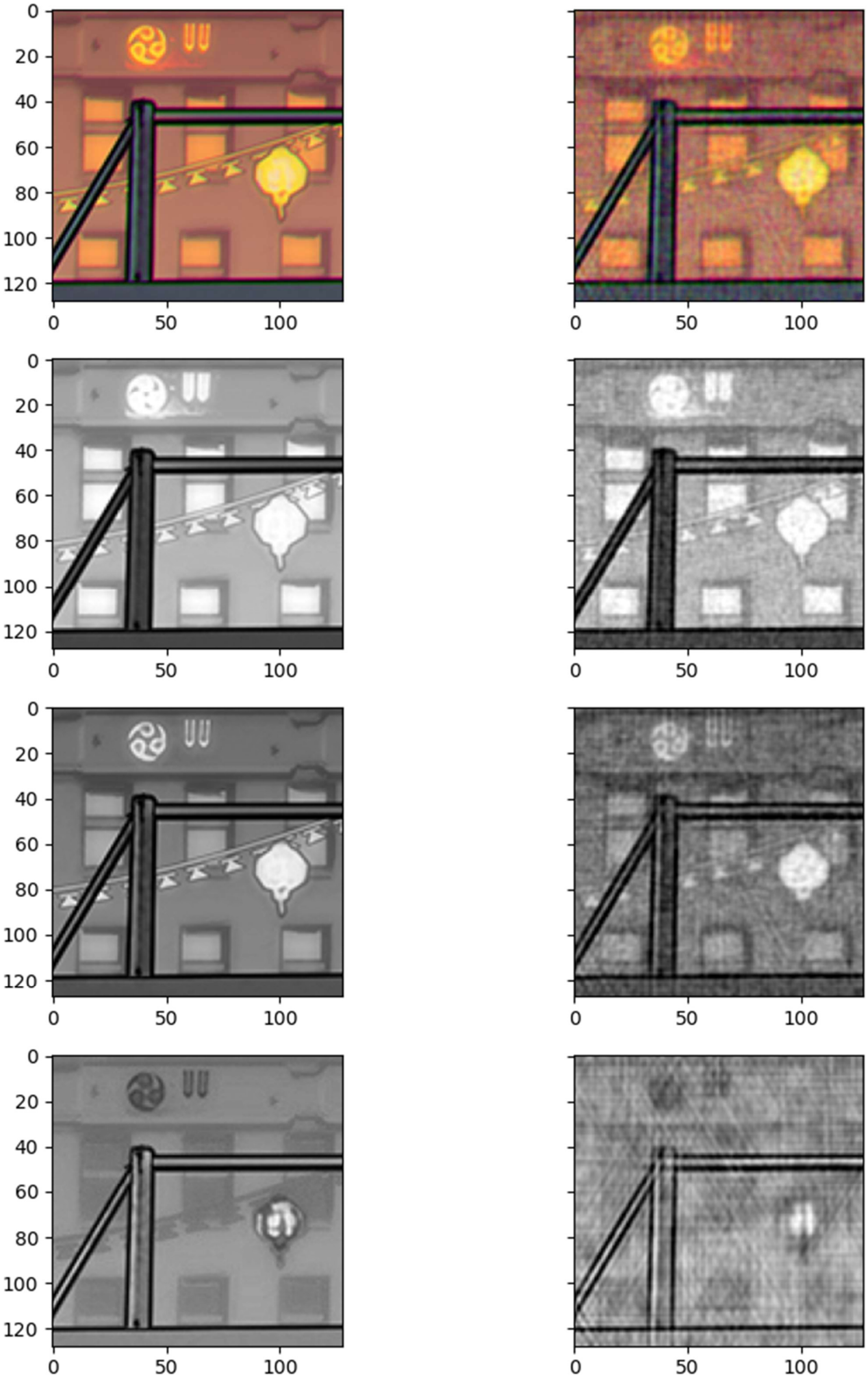
Subsampling 4:4:4, lum, chrom 50%, 256x256



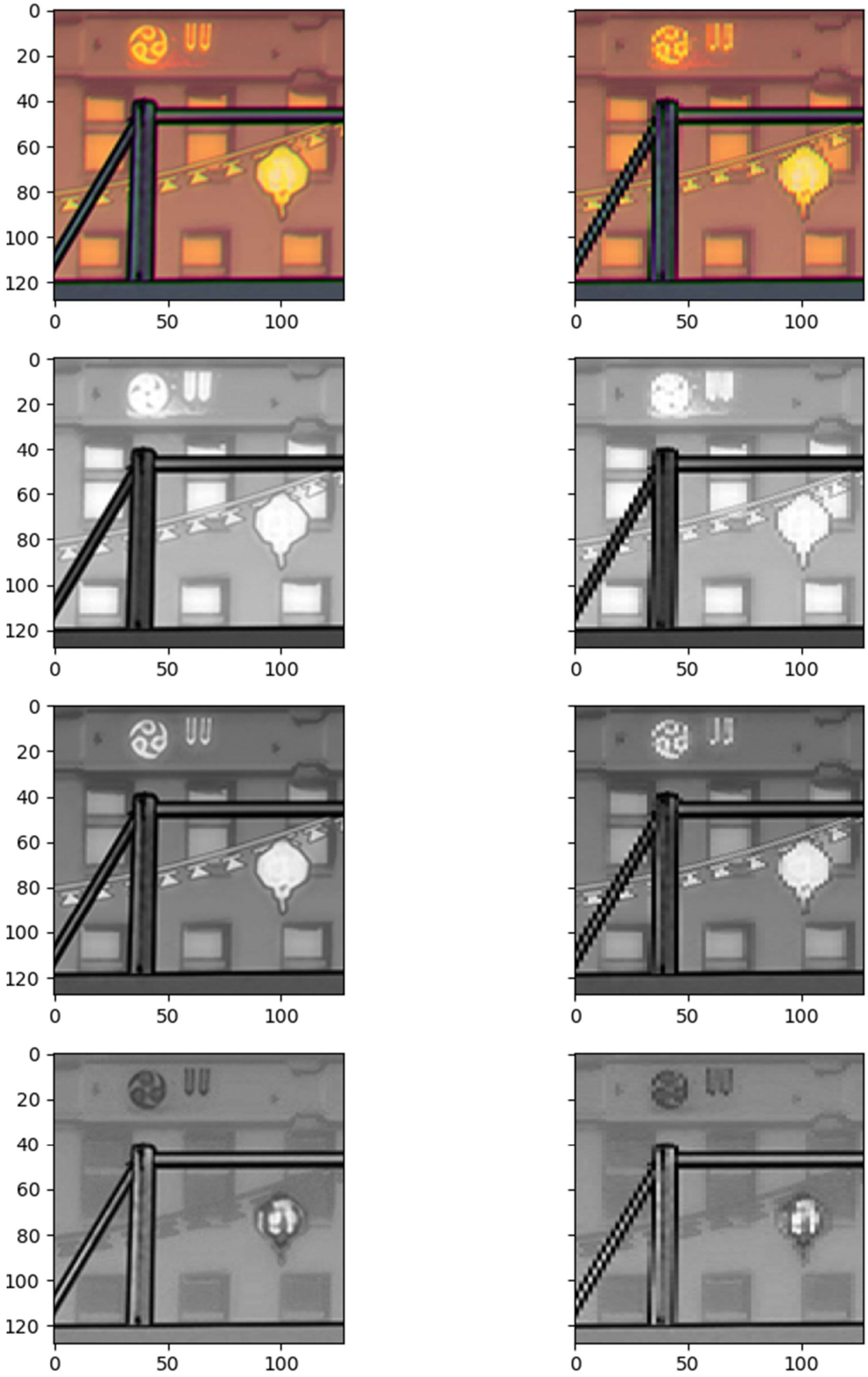
Subsampling 4:2:2, lum, chrom 50%, 256x256



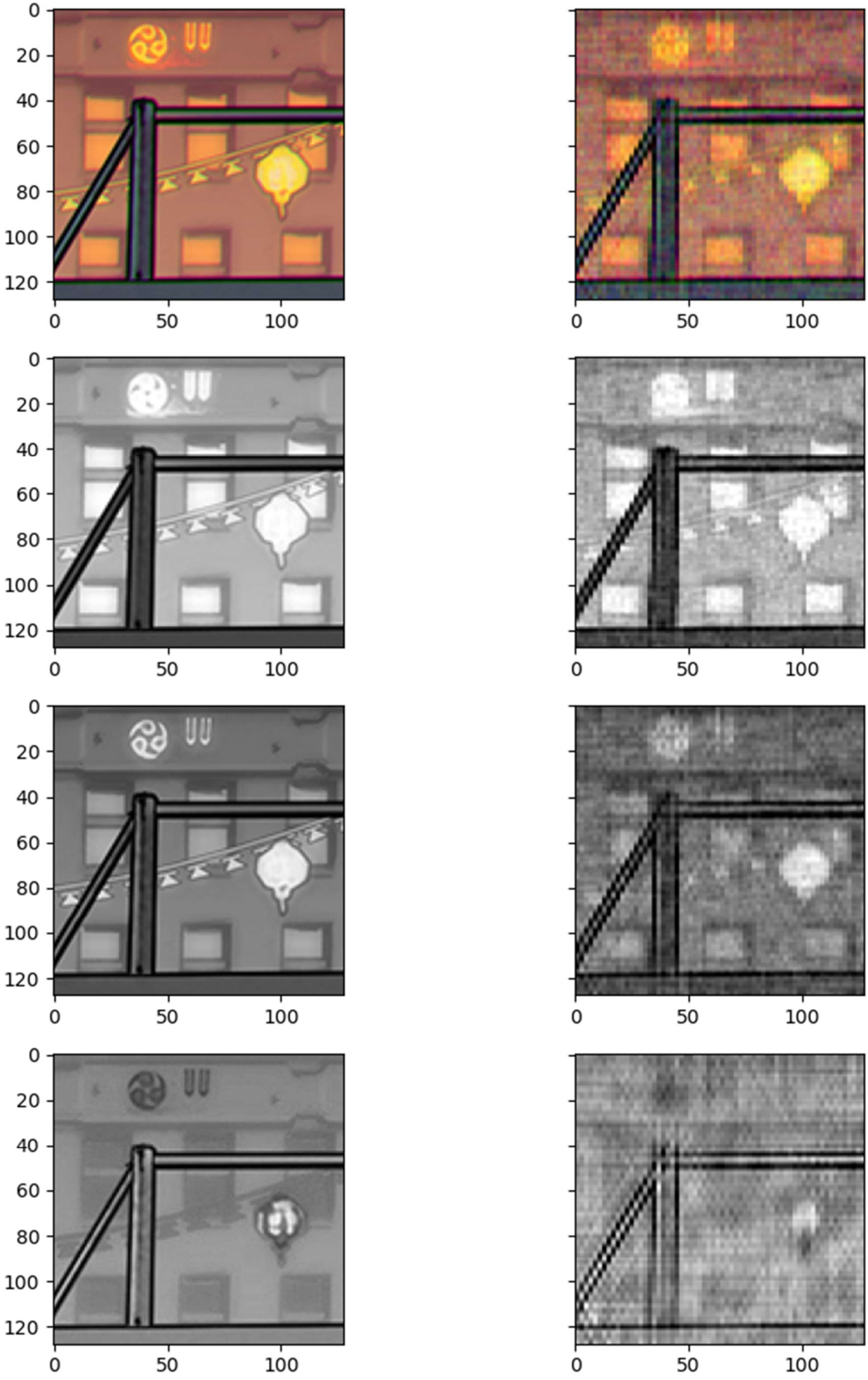
Subsampling 4:2:2, lum, chrom 100%, 256x256



Subsampling 4:4:4, lum, chrom 50%, 128x128



Subsampling 4:2:2, lum, chrom 100%, 128x128



Subsampling 4:2:2, lum, chrom 50%, 128x128

Wnioski:

Subsampling 4:2:2 powoduje delikatną pikselację obrazu, z kolei kompresja luminancji i chrominancji powoduje wyraźny „szum”. Najlepiej wyglądają obrazy poddane jedynie subsamplingowi, szczególnie gdy na obrazie nie ma wiele szczegółów. Kompresja luminancji i chrominancji tak mocno zniekształca obraz, że dodanie do niego subsamplingu jest najczęsciej ledwo zauważalne. Kompresja lum i chrom nie zniekształca szczegółów ale nakłada na cały obraz silny szum który jest bardzo zauważalny nawet gdy obraz jest jednolity.