

Systemy multimedialne

Daniel Brzezicki (bd46477)

Lab – 04.05.2022

Zadanie 1

Do zaimplementowania (0.7 pkt):

Algorytm realizujący nasz uproszczony algorytm JPEG dający wybór pomiędzy 4 kombinacjami poniższych kombinacji:

- Redukcję chrominancji - wybór pomiędzy 4:4:4 a 4:2:2
- Wybór tablicy kwantyzującej lub pominięcie jej (albo zastąpienie jej tablicą jedynek)

```
def chromaSubsample(data, params):  
    y,x = data.shape  
  
    result=np.empty((y, int(x/2)))  
    if params=="4:2:2":  
        for _y in range(0,y):  
            for _x in range(0,x,2):  
                result[_y][int(_x/2)]=data[_y][_x]  
    else:  
        return data  
  
    return result
```

```
def chromaDesubsample(data, params):  
    y,x = data.shape  
  
    result=np.empty((y, x*2))  
    if params=="4:2:2":  
        for _y in range(0,y):  
            for _x in range(0,x):  
                result[_y][_x*2]=data[_y][_x]  
                result[_y][_x*2+1]=data[_y][_x]  
    else:  
        return data  
  
    return result
```

```

def compress(sourceIm,params,t1,t2,t3):
    Y, Cr, Cb = cv2.split(sourceIm)

    data = DATA()

    data.Y = compressHelper(Y,params,t1,False)
    data.Cr = compressHelper(Cr,params,t2)
    data.Cb= compressHelper(Cb,params,t3)

    data.Y=RLE(data.Y)
    data.Cb =RLE(data.Cb)
    data.Cr=RLE(data.Cr)

    return data

def compressHelper(data, params, tab, chroma=True):

    cs=chromaSubsample(data,params,chroma)
    cs = cs.astype(int)-128
    cs_y,cs_x = cs.shape
    result=np.zeros(cs_y*cs_x)

    idx=0
    for _y in range(0,cs_y,8):
        for _x in range(0,cs_x,8):
            dct = dct2(cs[_y:_y+8,_x:_x+8])
            quant=(dct/tab).astype(int)#/tab
            result[idx:idx+64]=zigzag(quant)
            idx+=64
    return result

```

```

def decompress(data, params,t1,t2,t3):

    Y = decodeRLE(data.Y)
    Cr = decodeRLE(data.Cr)
    Cb = decodeRLE(data.Cb)

    Y=decompressHelper(Y,params,t1,False)
    Cr=decompressHelper(Cr,params,t2)
    Cb=decompressHelper(Cb,params,t3)

    return np.dstack([Y,Cr,Cb]).astype(np.uint8)

def decompressHelper(data,params,tab,chroma=True):
    result = initData(params,data.shape[0],chroma)

    y,x=result.shape
    for idx, i in enumerate(range(0,data.shape[0],64)):
        dezigzagged=zigzag(data[i:i+64])
        dequantized=dezigzagged*tab
        _x=(idx*8)%x
        _y=int((idx*8)/x)*8
        result[_y:_y+8, _x:_x+8]=idct2(dequantized)+128

    result= chromaDesubsample(np.clip(result,0,255).astype(np.uint8),params,chroma)
    return result

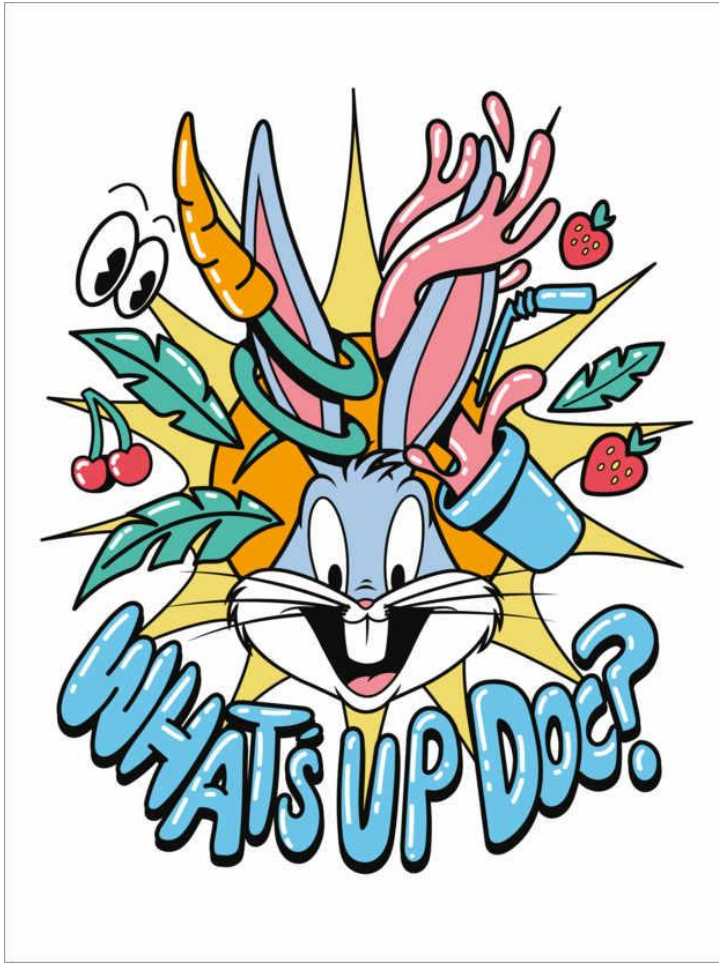
```

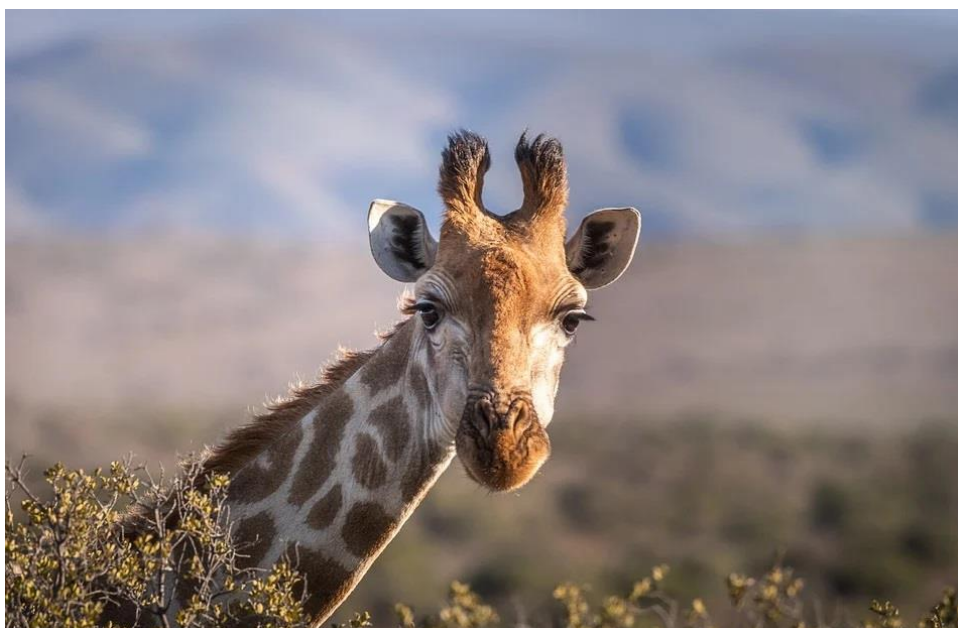
Zadanie 2

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. Wycinki najlepiej, żeby były jednego rozmiaru najlepiej 128x128 lub 256x256.

Wybrane zdjęcia:

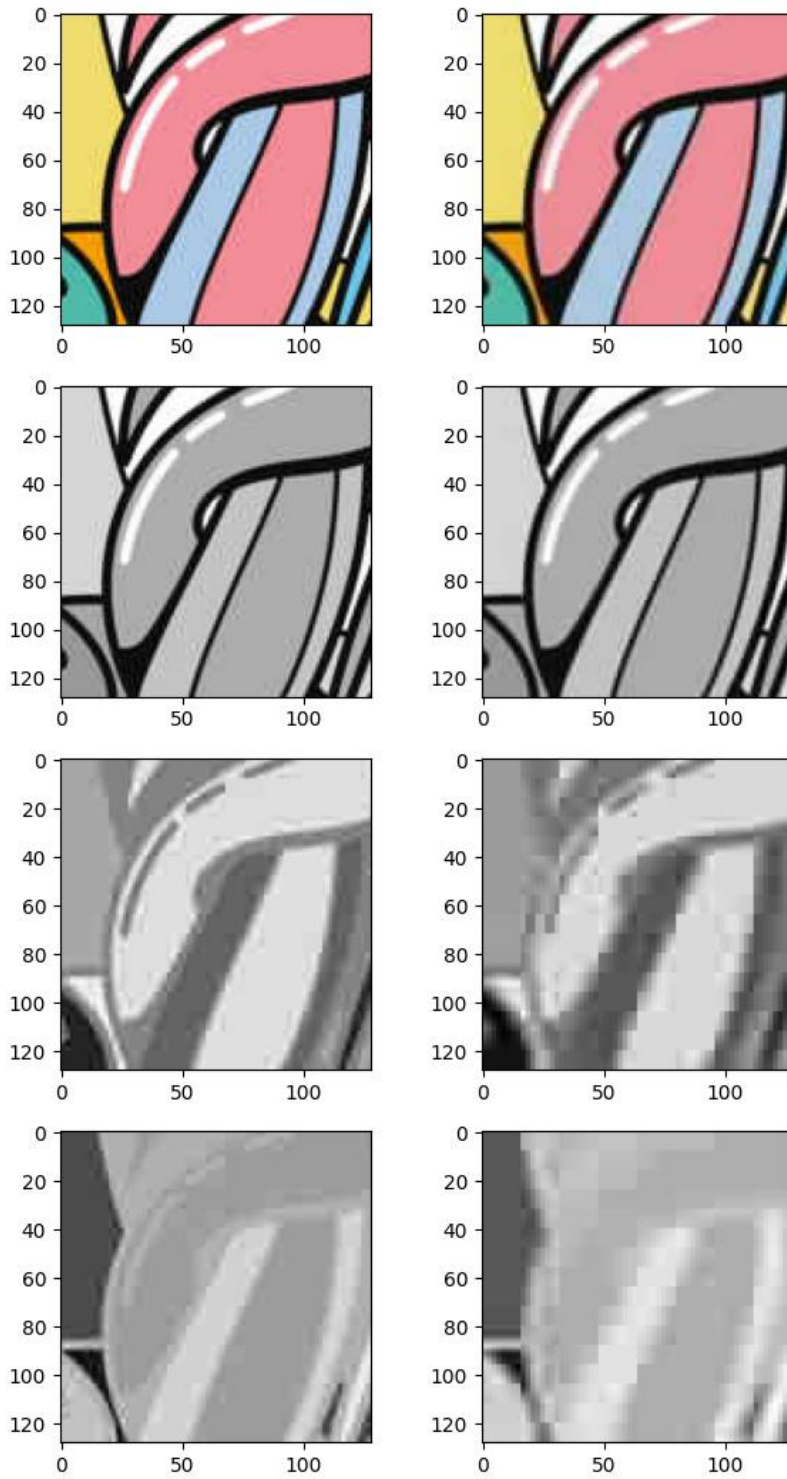




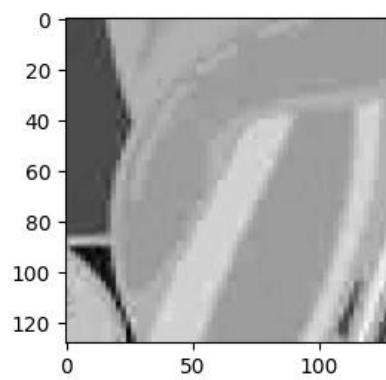
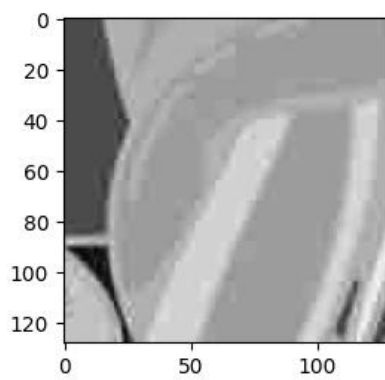
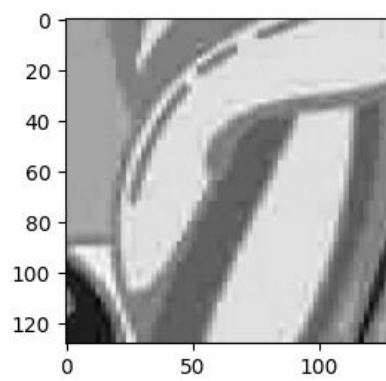
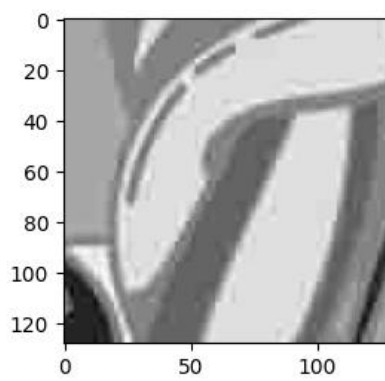
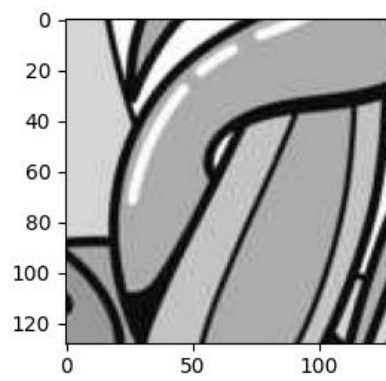
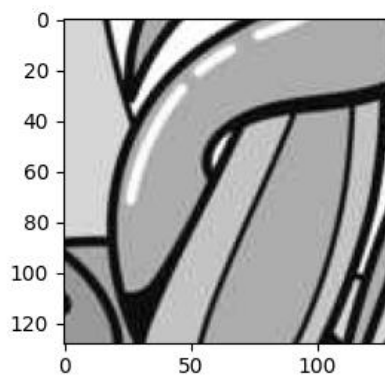
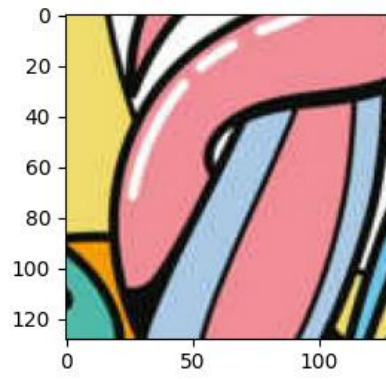
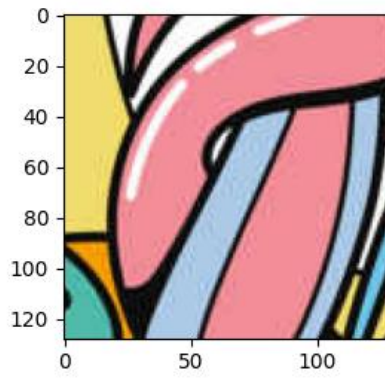
Krótkie wnioski:

Subsampling znacznie wpływa na wycinki, pojawia się pikselizacja. Przy 4:2:2 widac znaczącą pikselizację, czego nie można powiedzieć o 4:4:4. Przy chrominacji obraz staje się nie wyraźny (lekko zaszumiony). Kompresja mocno przyczynia się do zniekształceń oryginalnych próbek. W każdym przedstawionym obrazku w tytule znajduje się rodzaj parametru subsamplingu oraz procent chrominancji (Kolejno na figure: oryginał, Y, Cr, Cb).

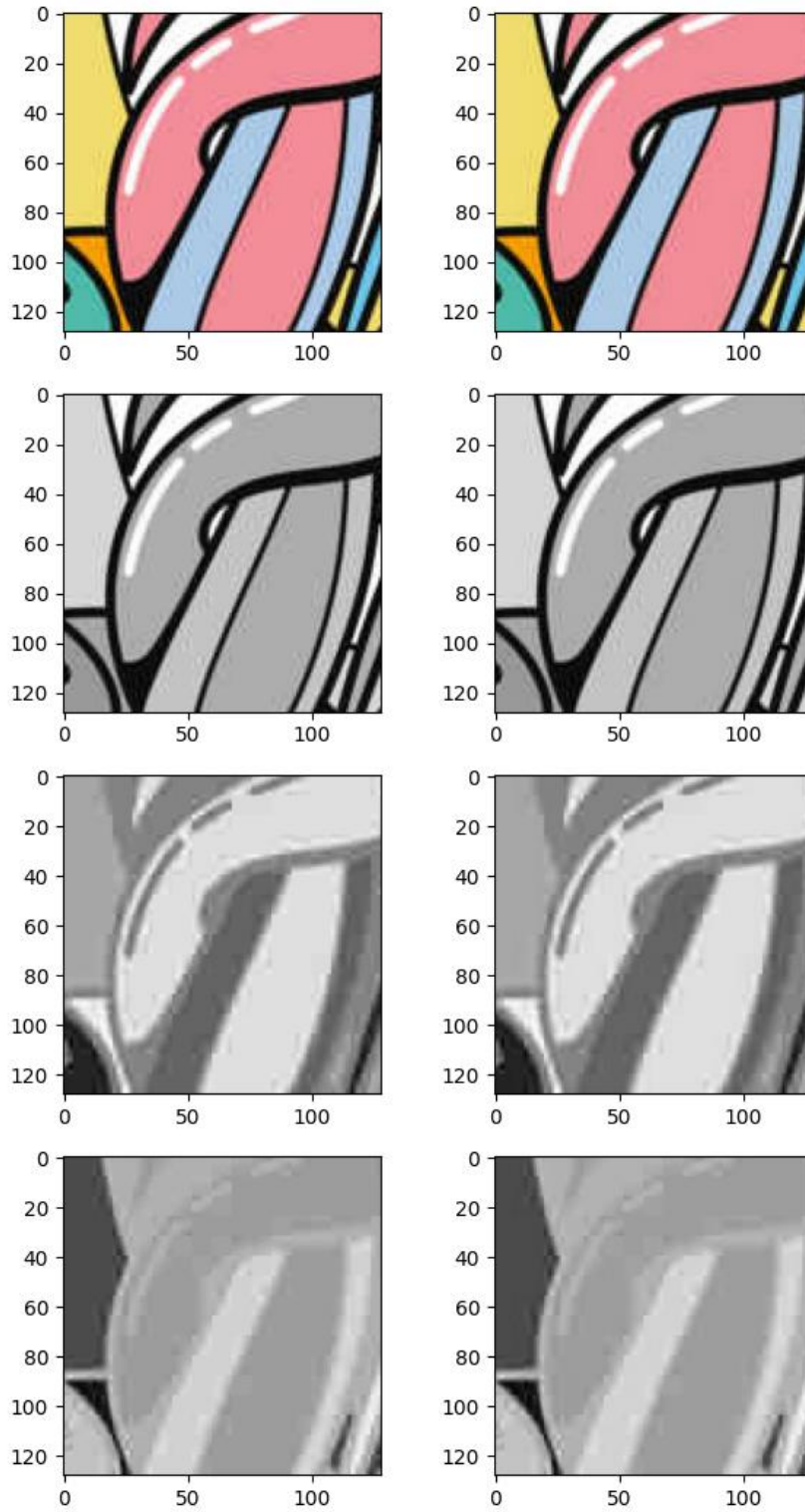
bug 300 200 4-2-2 50pro



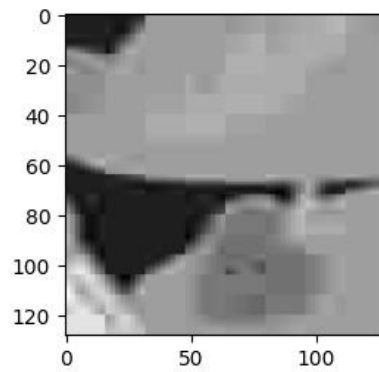
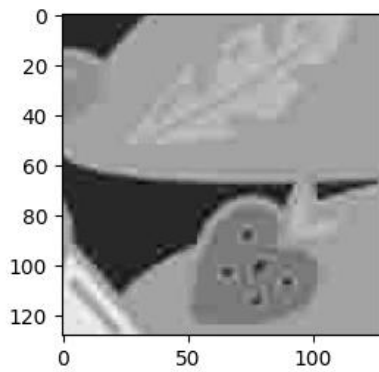
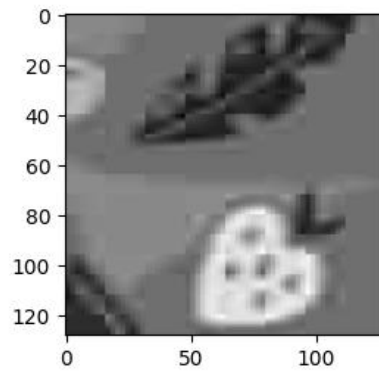
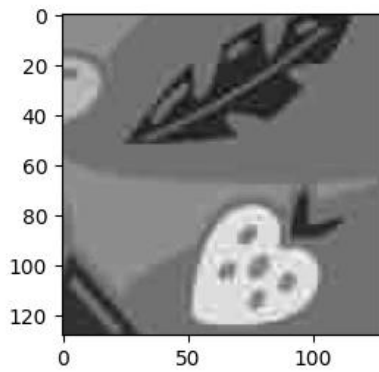
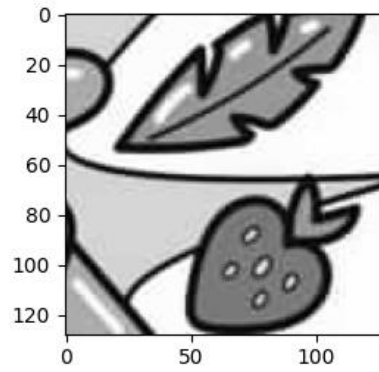
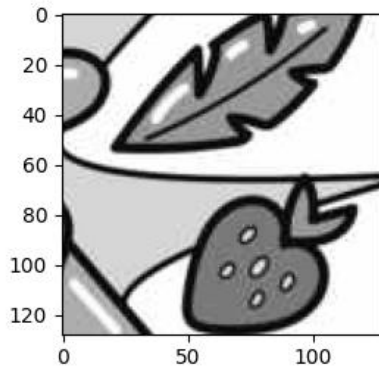
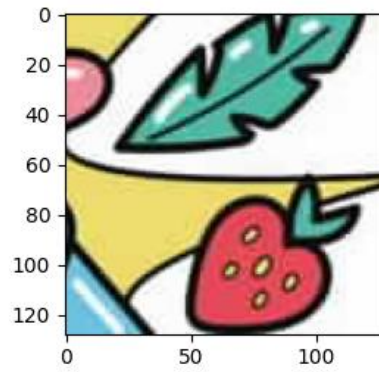
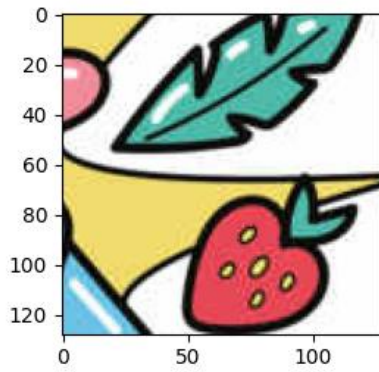
bug 300 200 4-2-2 100pro



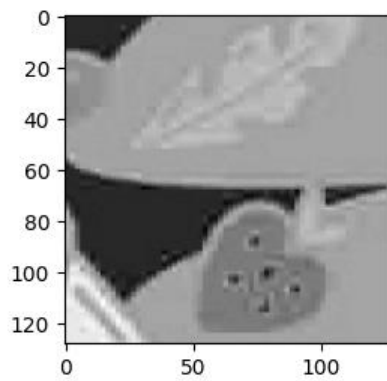
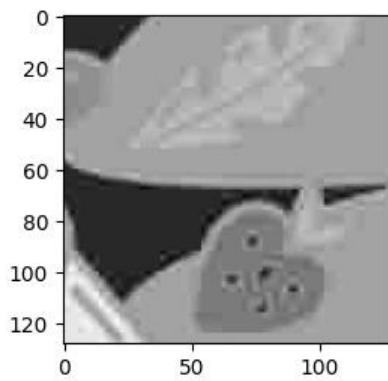
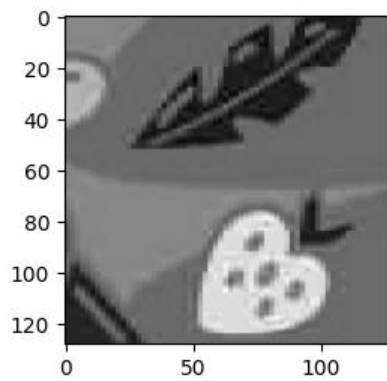
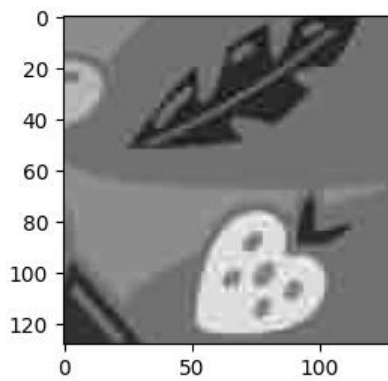
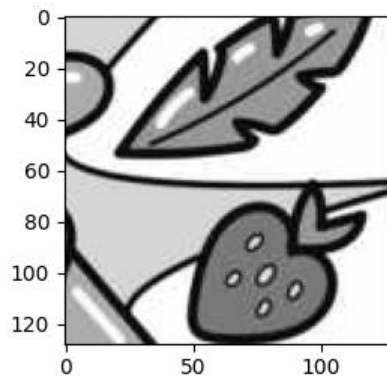
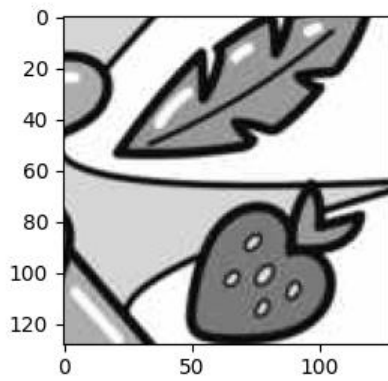
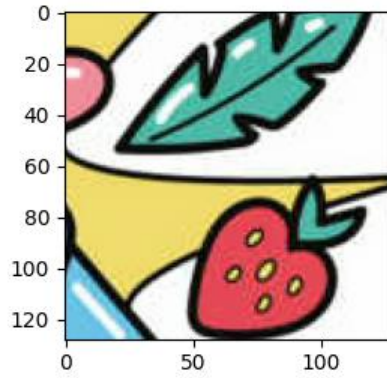
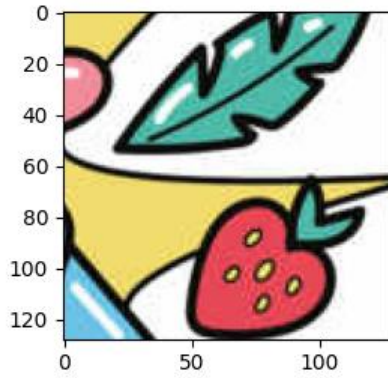
bug 300 200 4-4-4



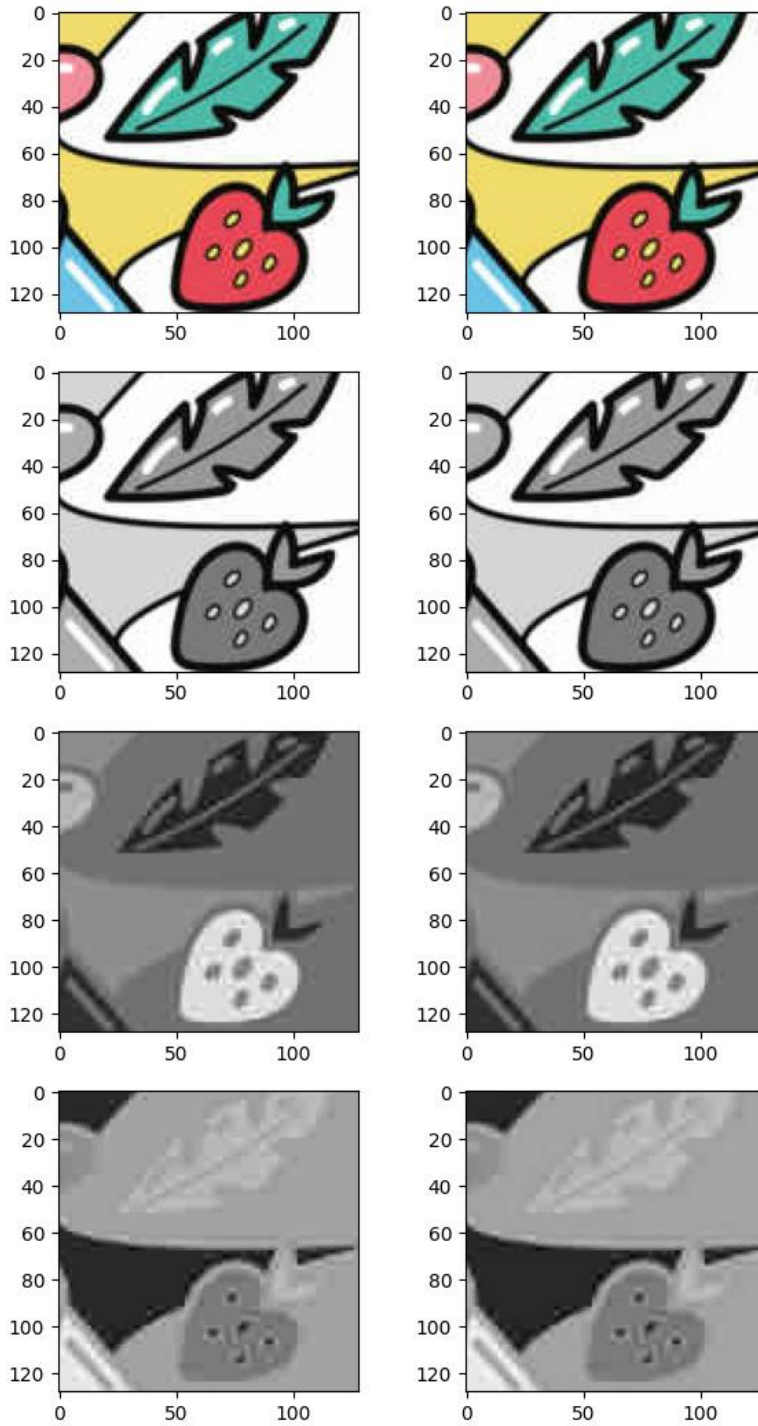
bug 450 300 4-2-2 50pro



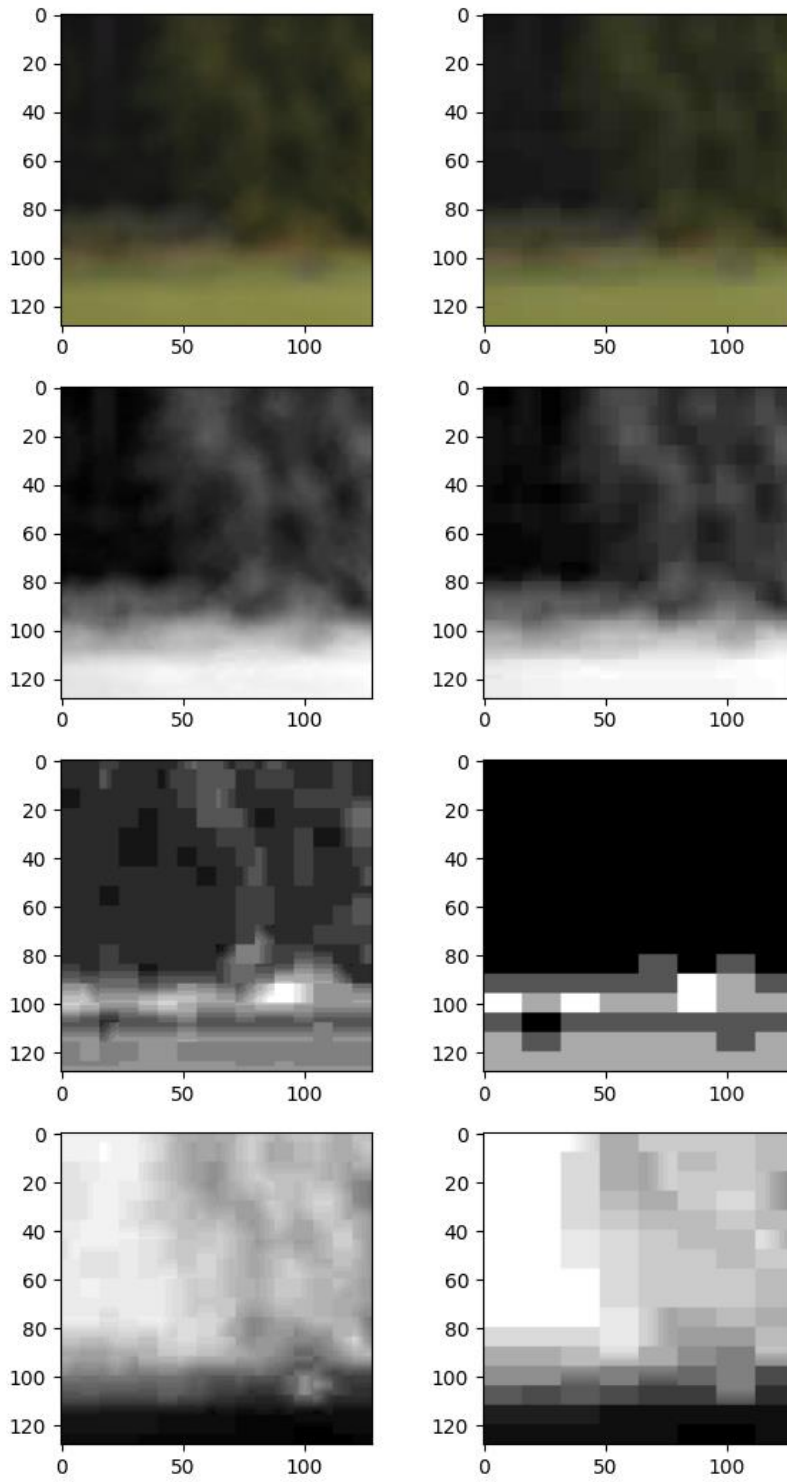
bug 450 300 4-2-2 100pro



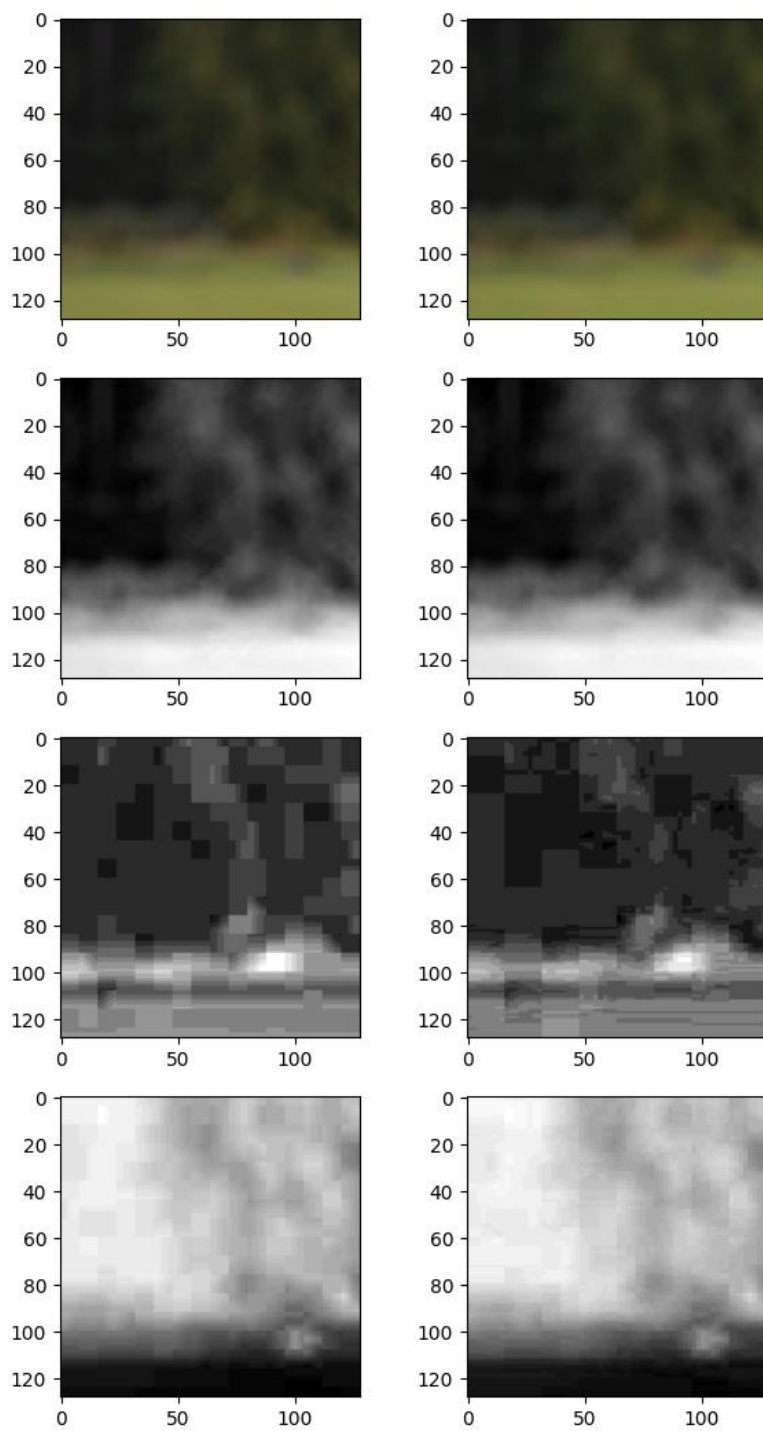
bug 450 300 4-4-4



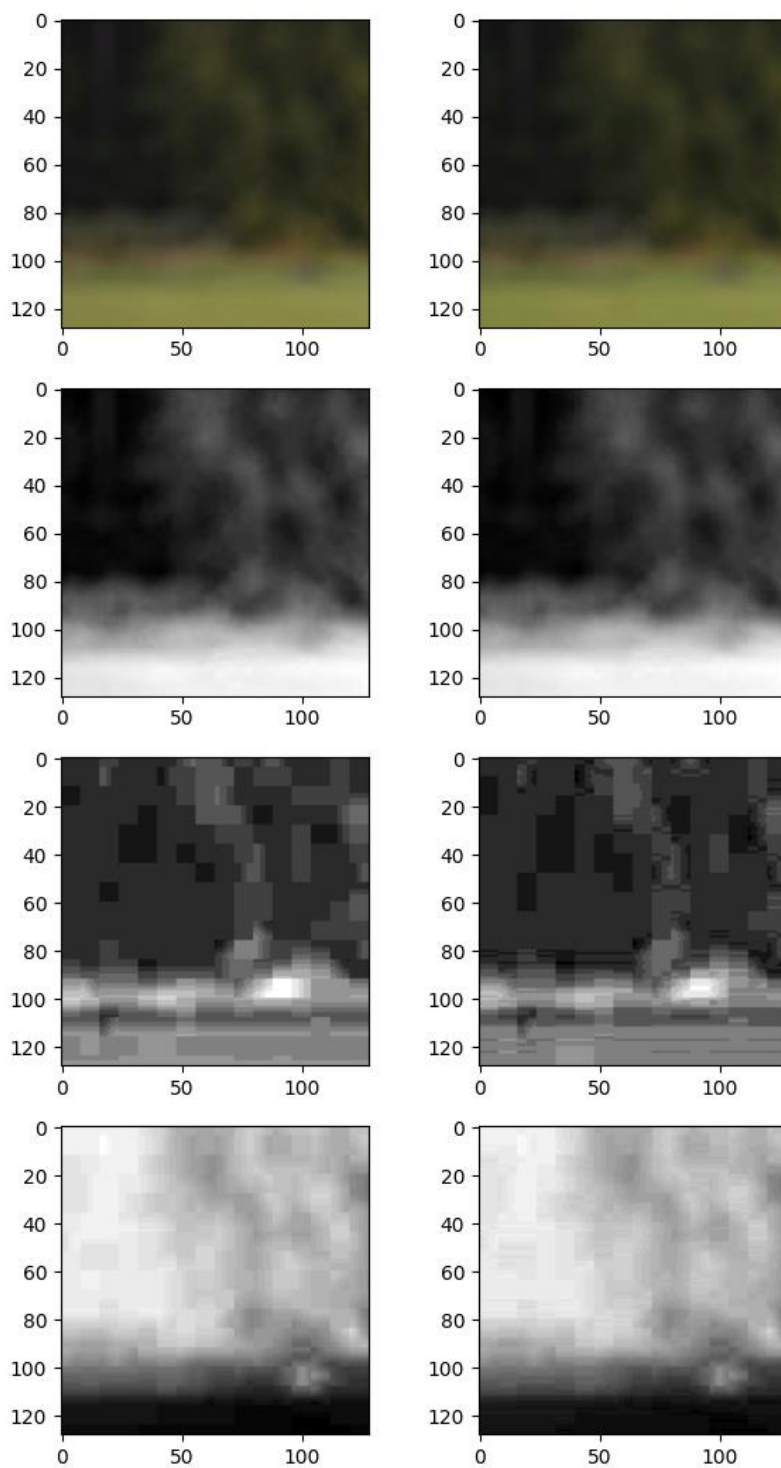
dog 0 100 4-2-2 50pro



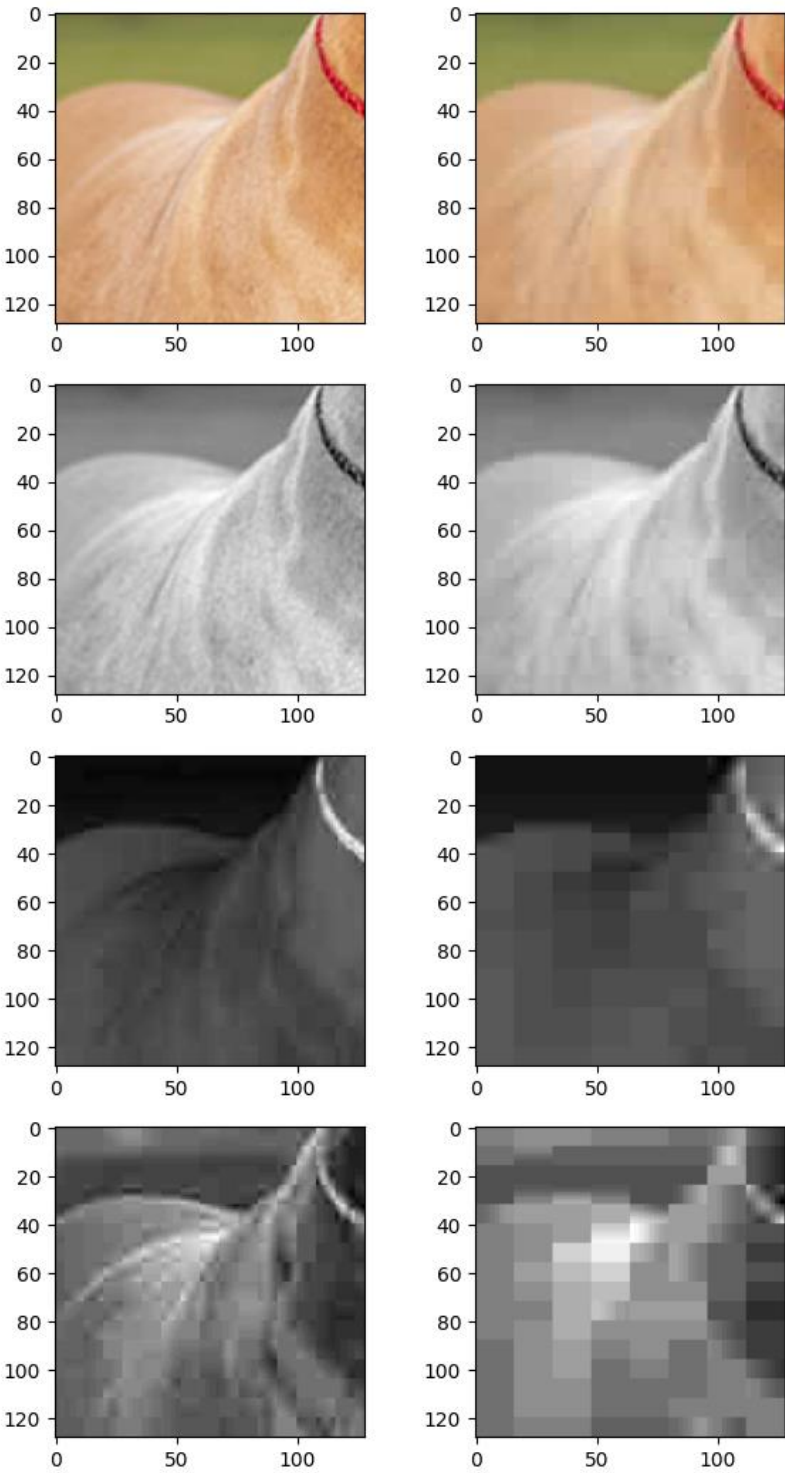
dog 0 100 4-2-2 100pro



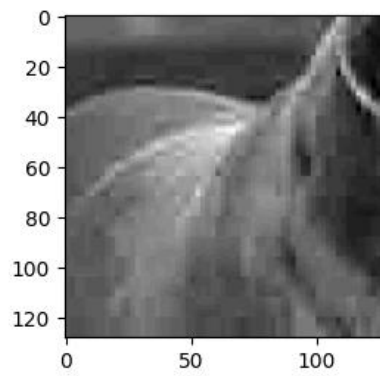
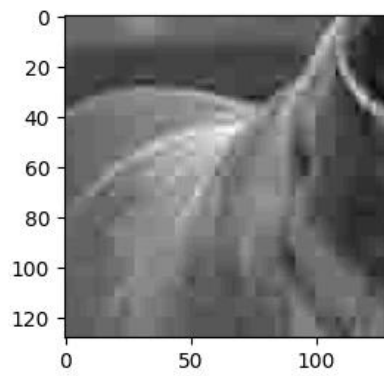
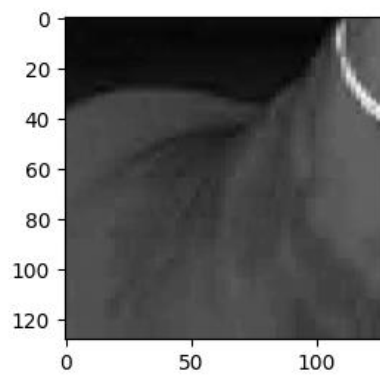
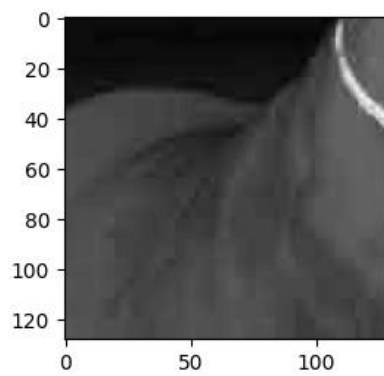
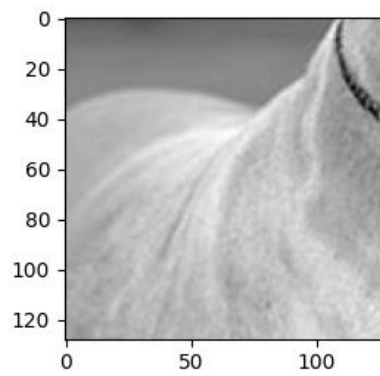
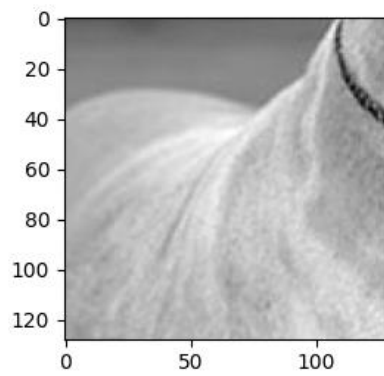
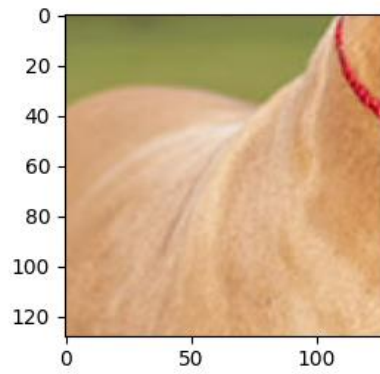
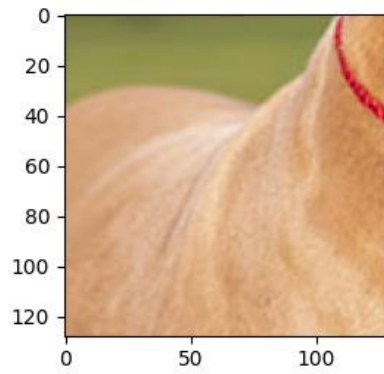
dog 0 100 4-4-4



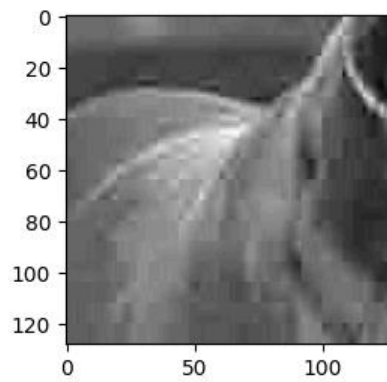
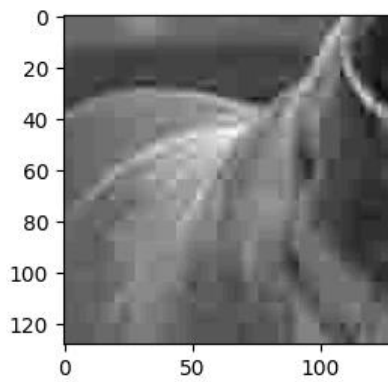
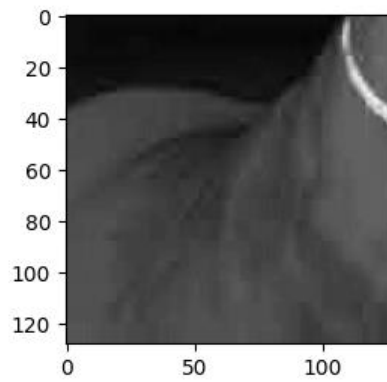
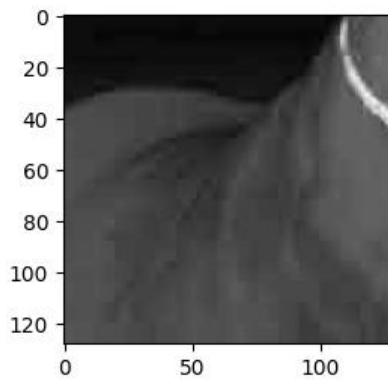
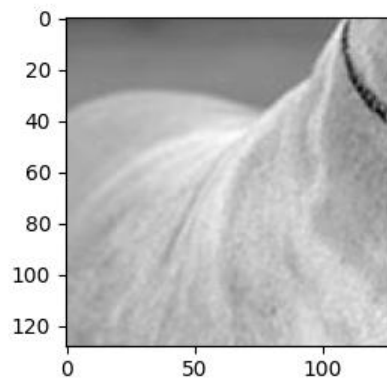
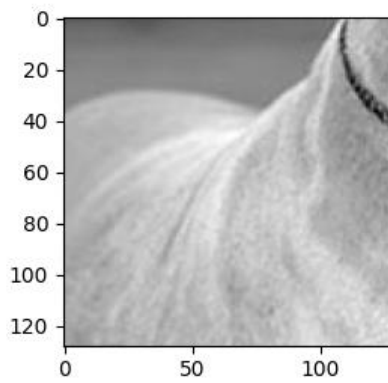
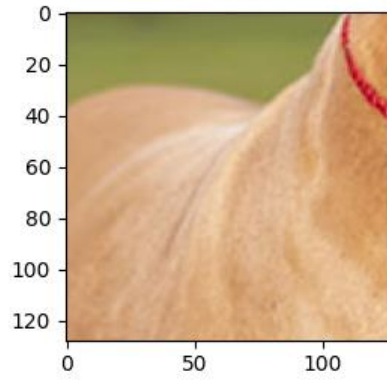
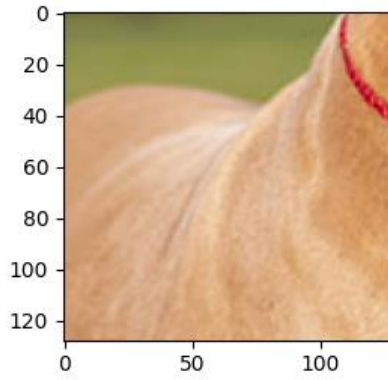
dog 300 200 4-2-2 50pro



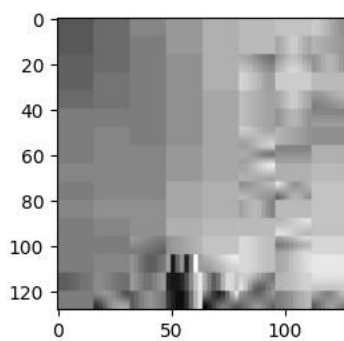
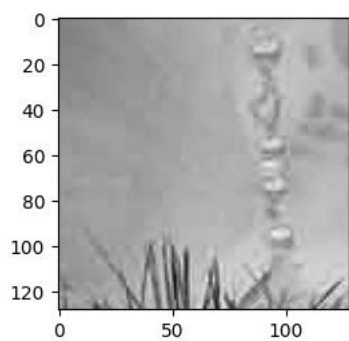
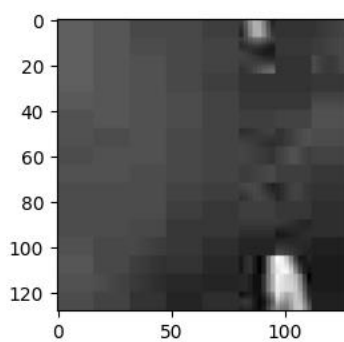
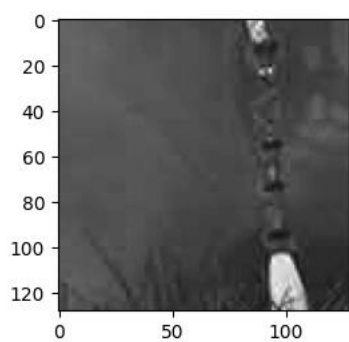
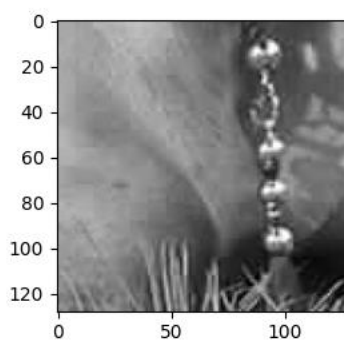
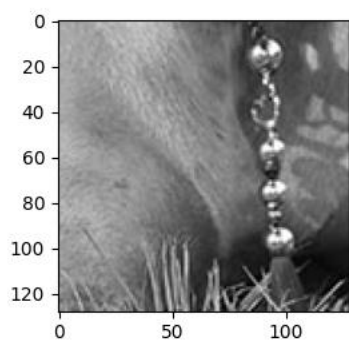
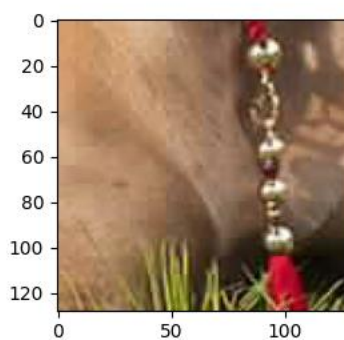
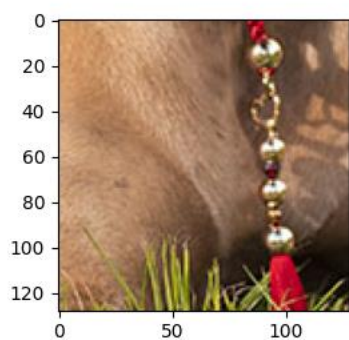
dog 300 200 4-2-2 100pro



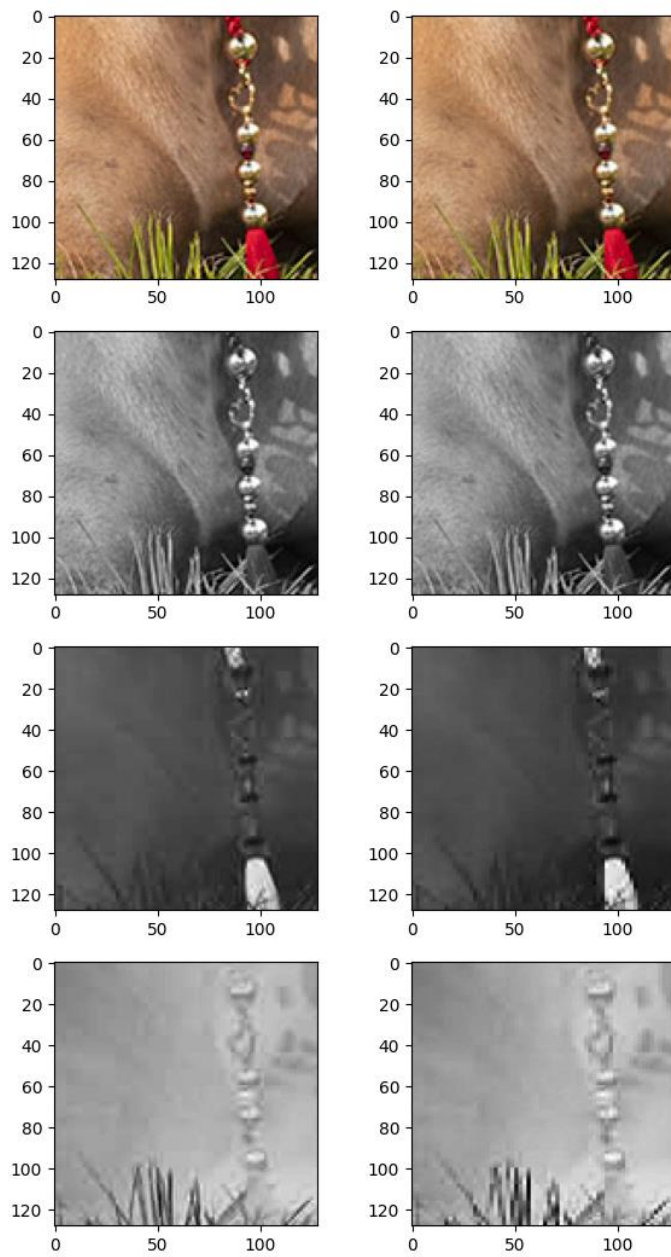
dog 300 200 4-4-4



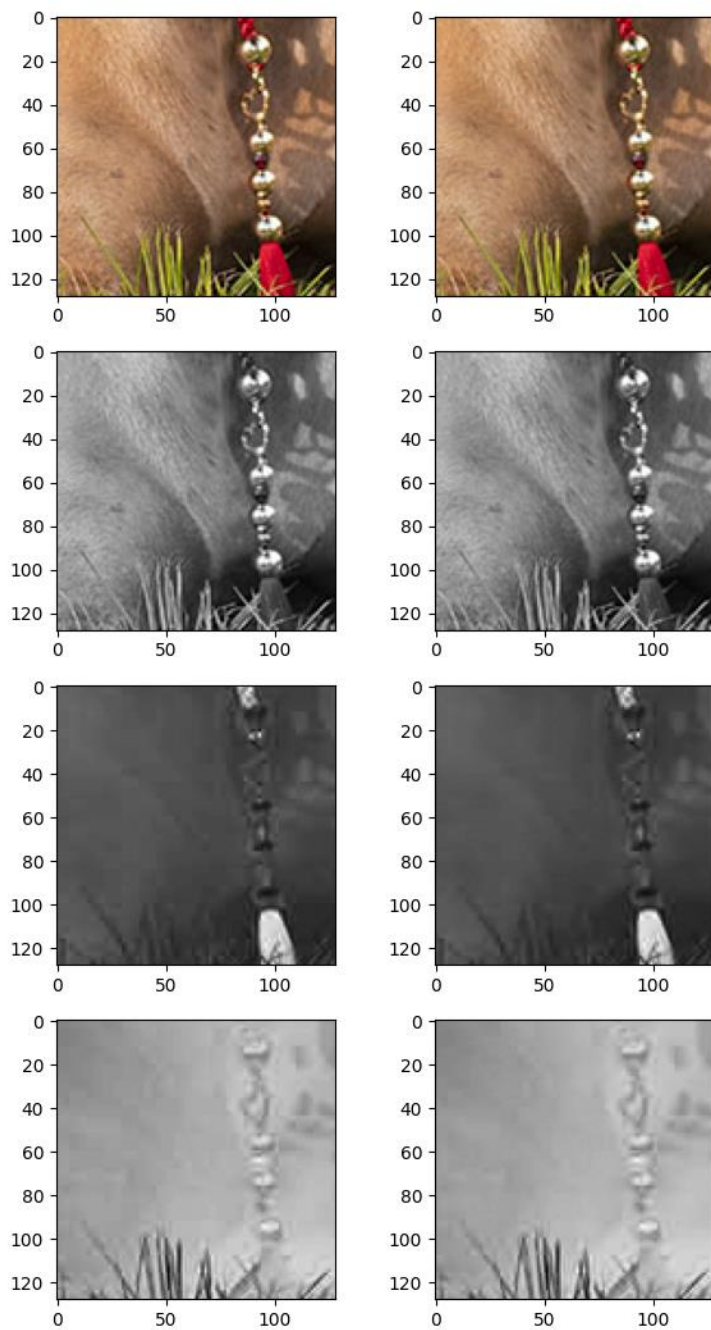
dog 450 300 4-2-2 50pro



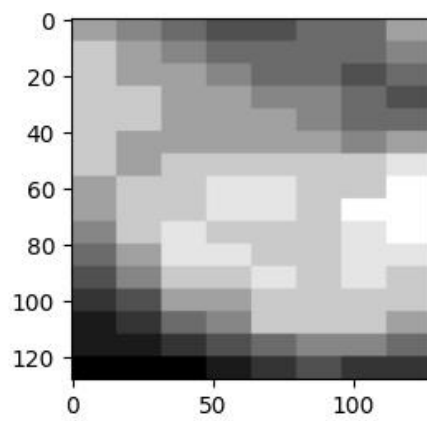
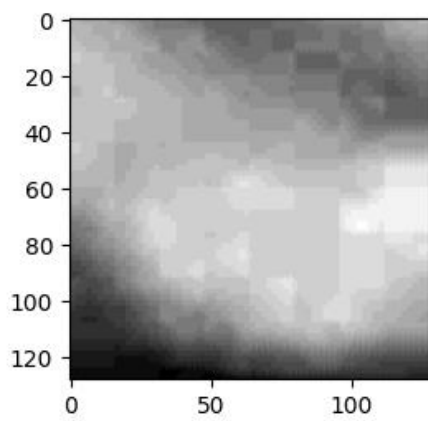
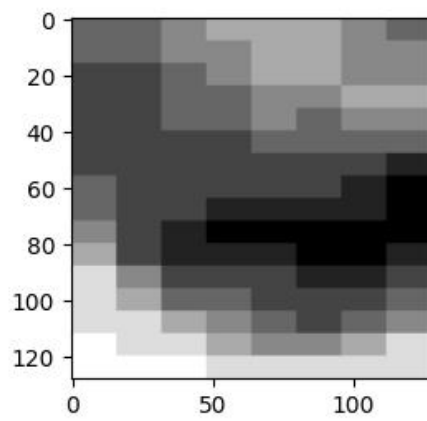
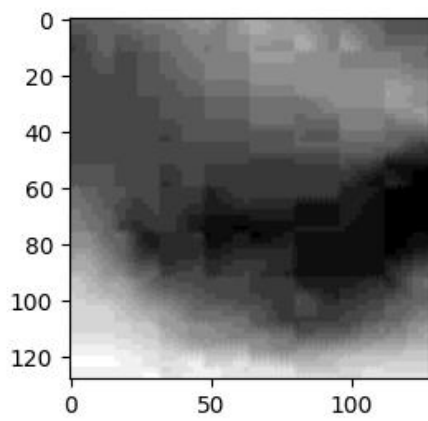
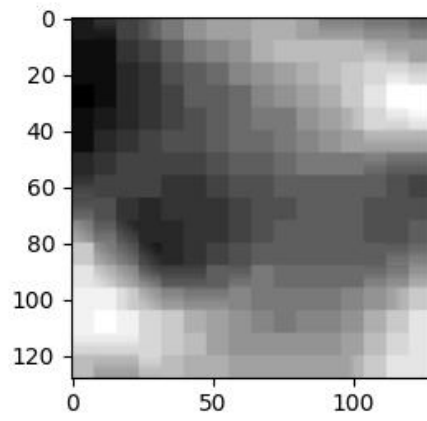
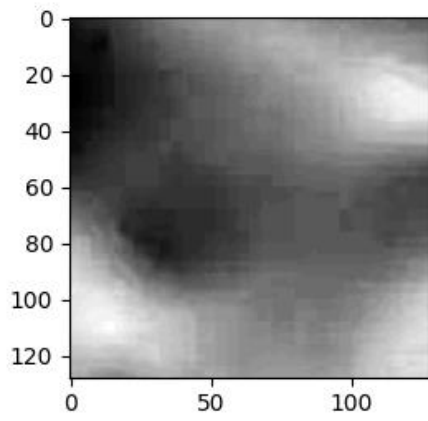
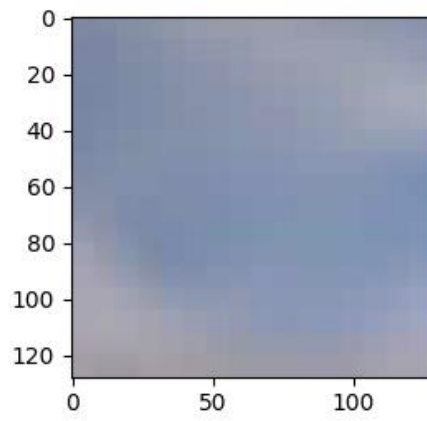
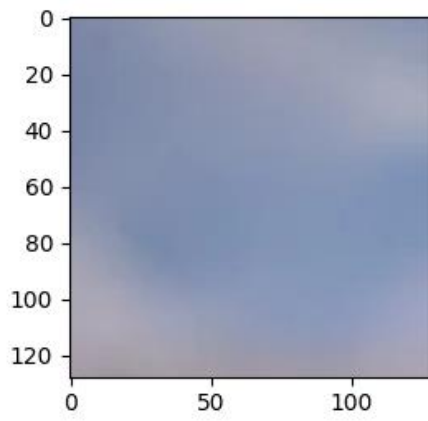
dog 450 300 4-2-2 100pro



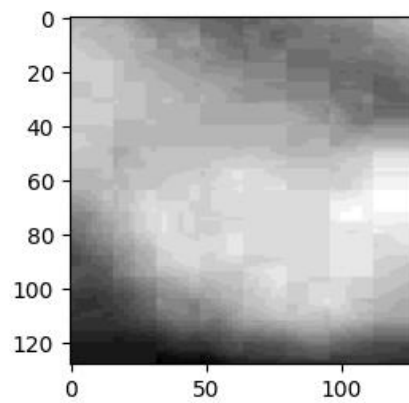
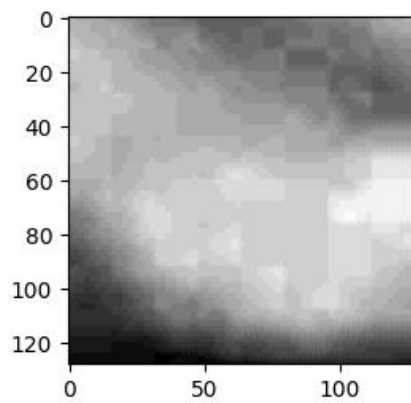
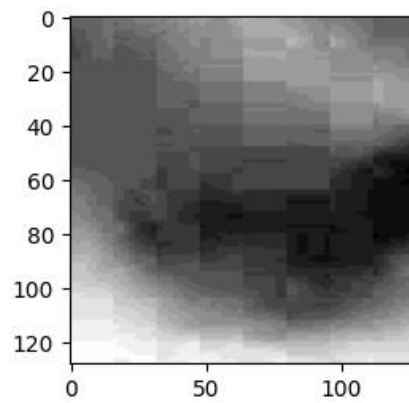
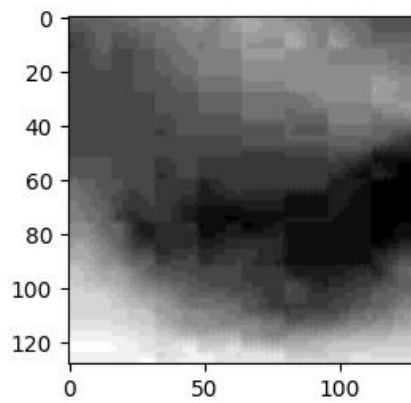
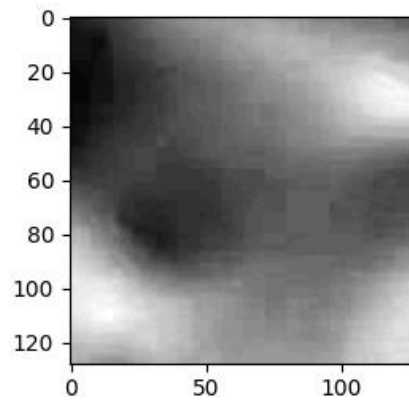
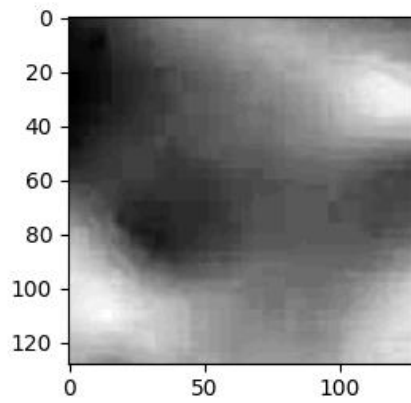
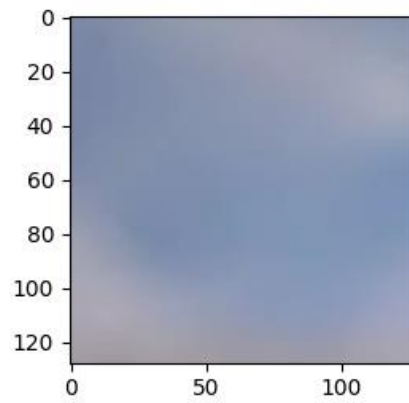
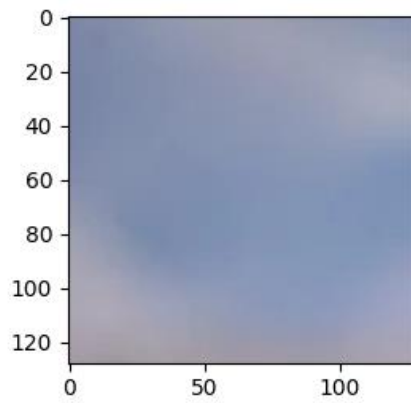
dog 450 300 4-4-4



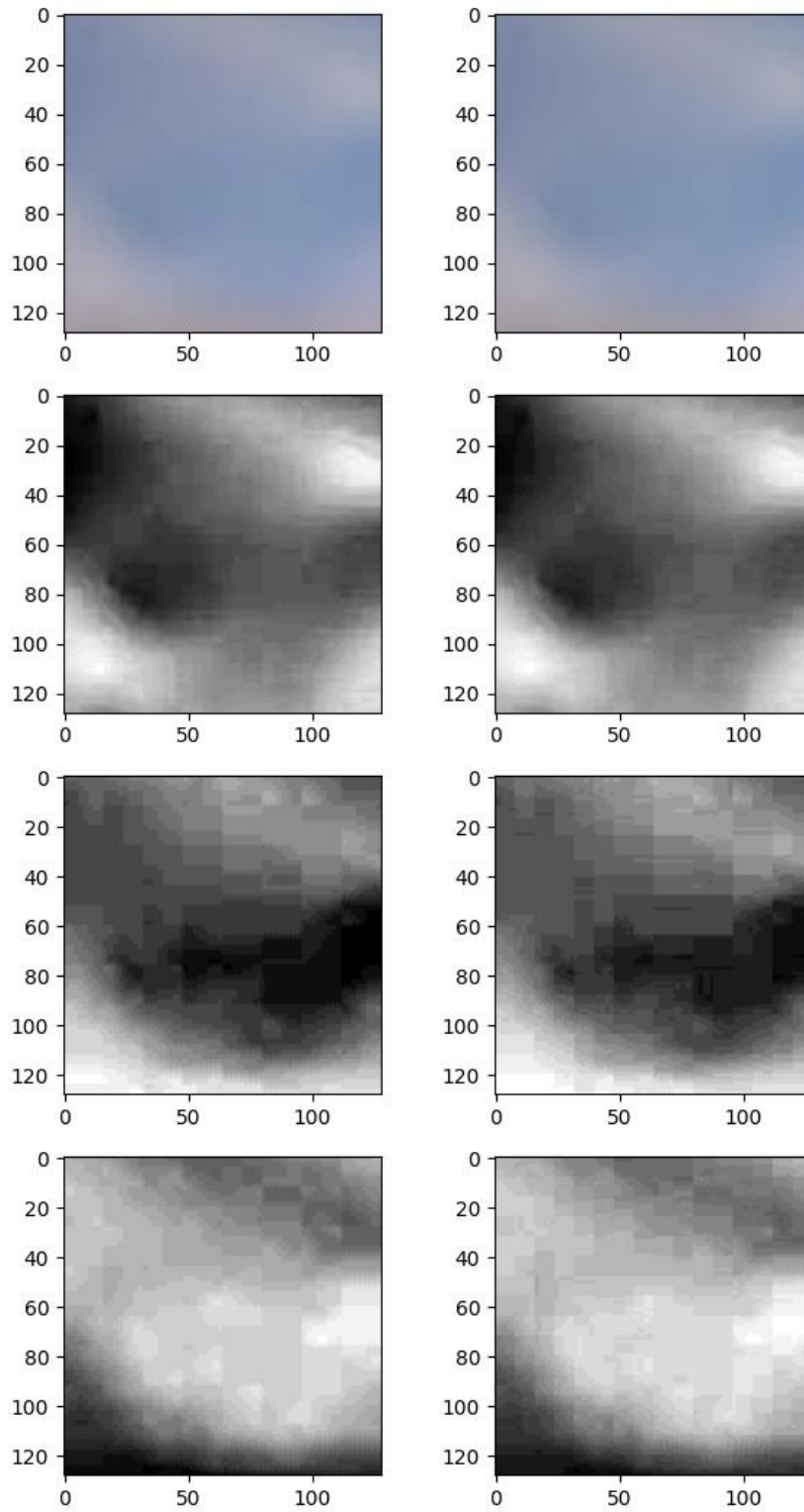
zyr 0 100 4-2-2 50pro



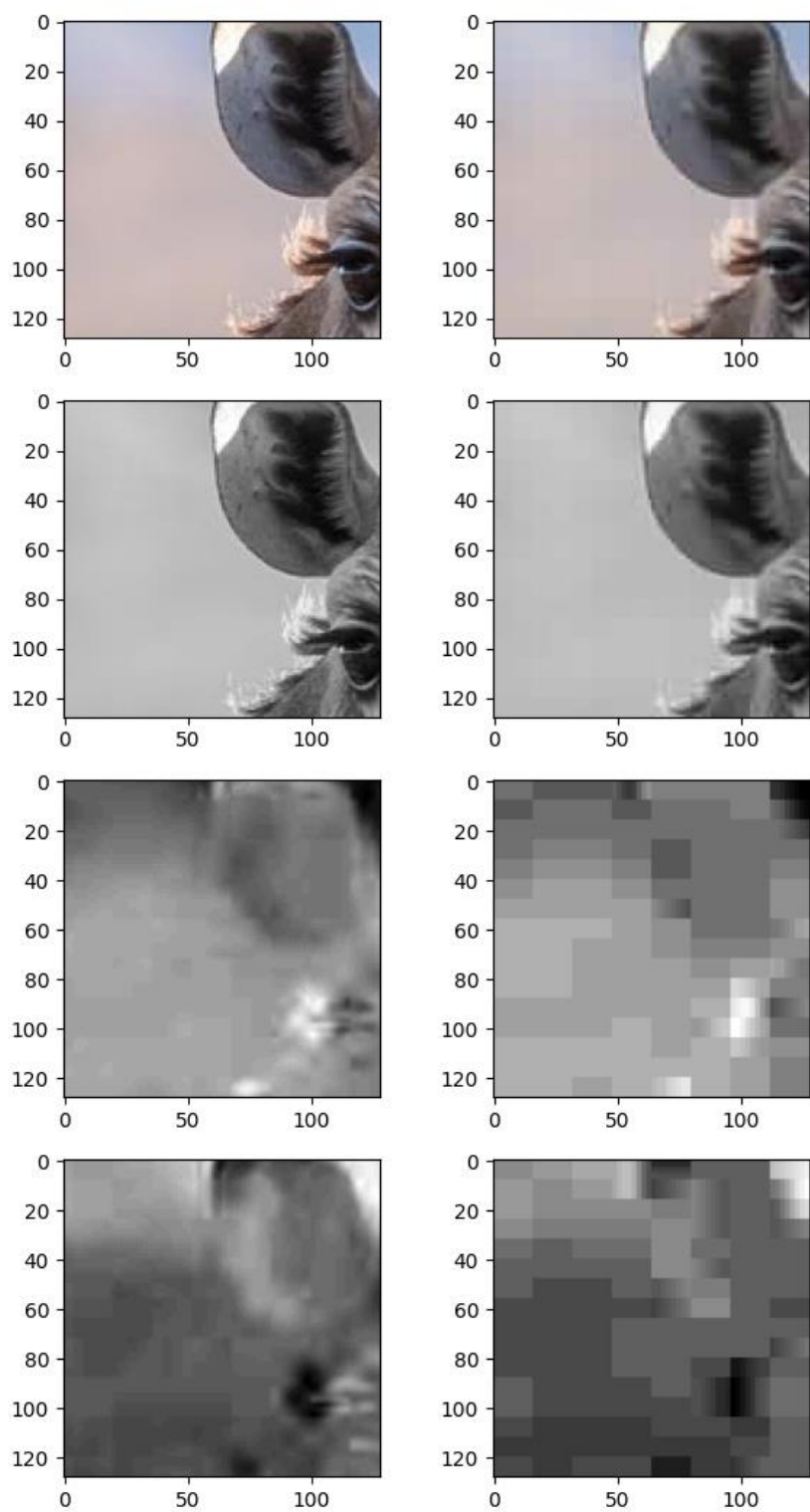
zyr 0 100 4-2-2 100pro



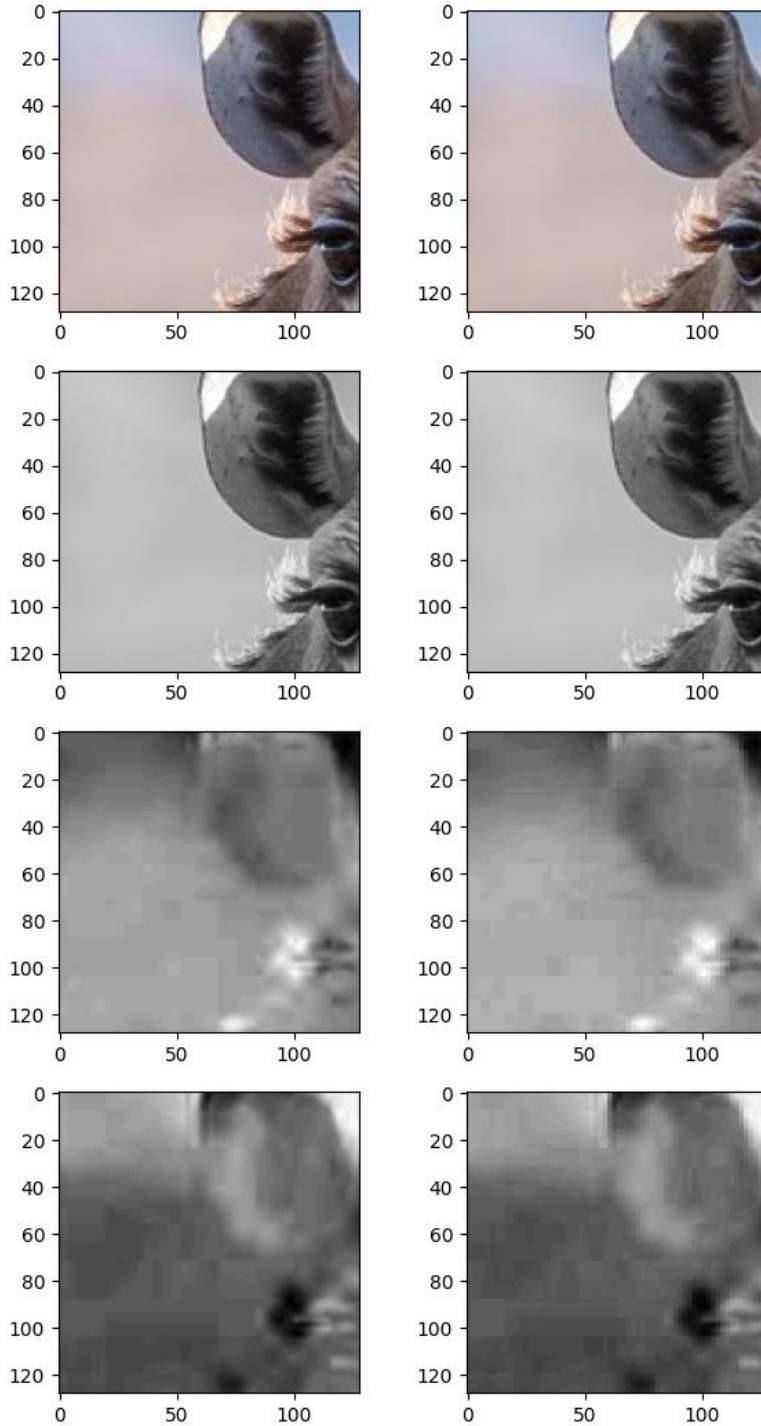
zyr 0 100 4-4-4



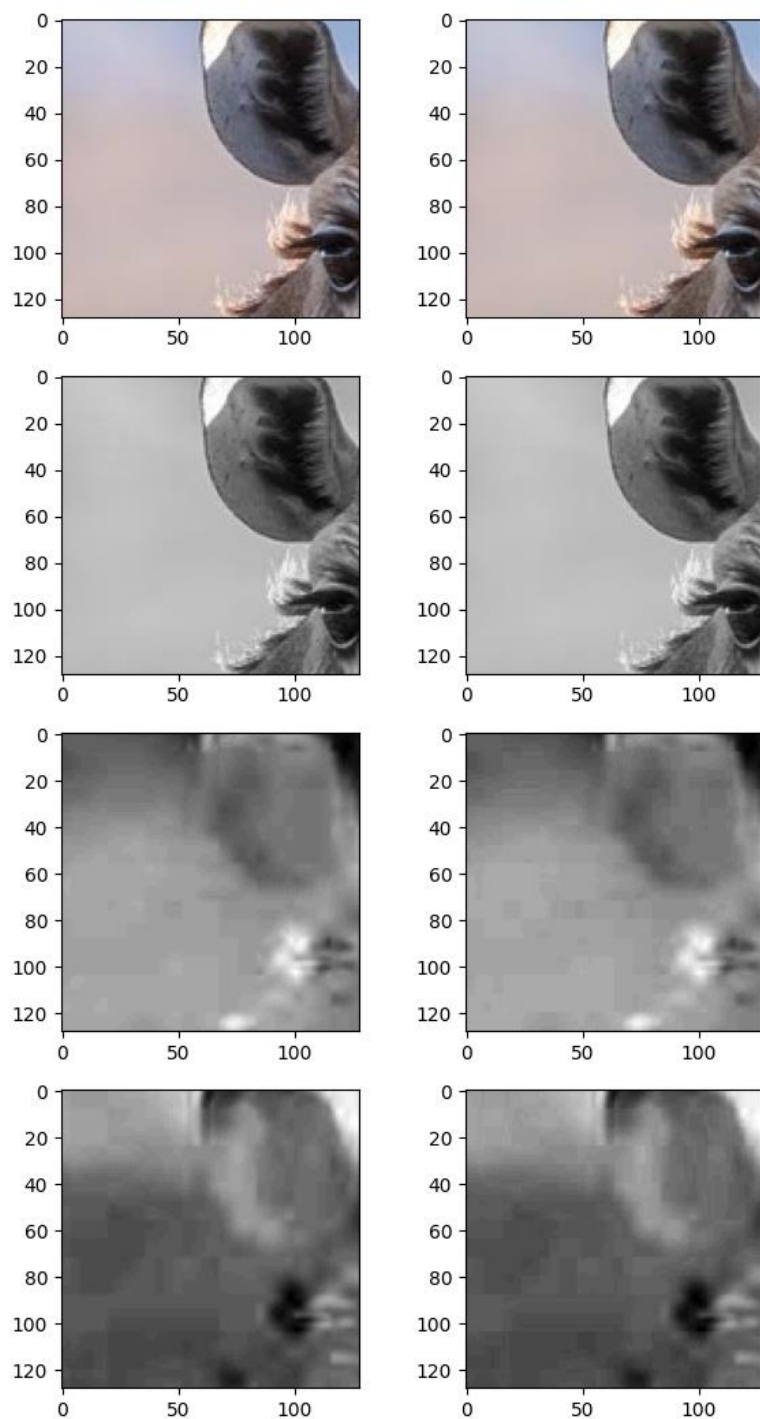
zyr 300 200 4-2-2 50pro



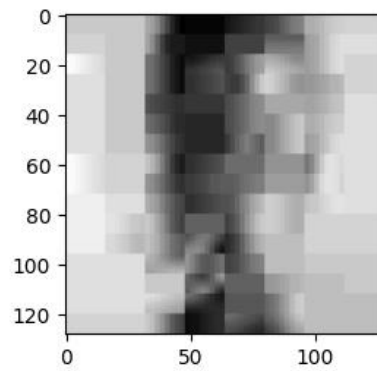
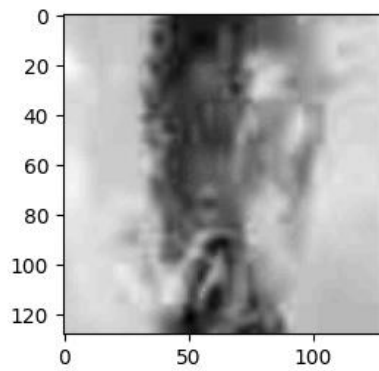
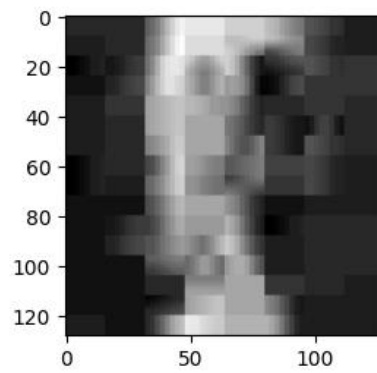
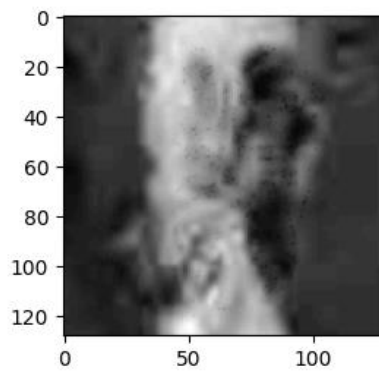
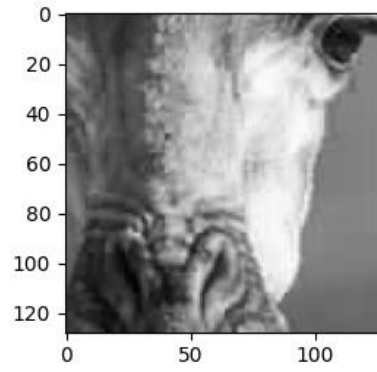
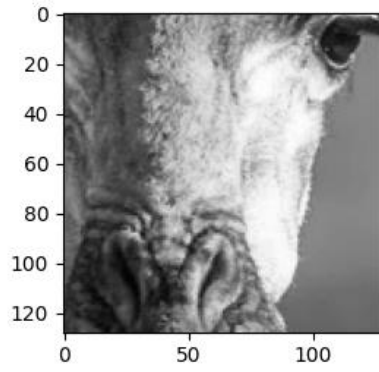
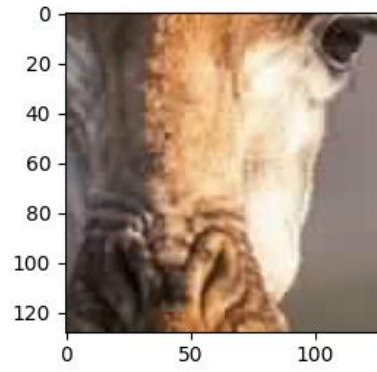
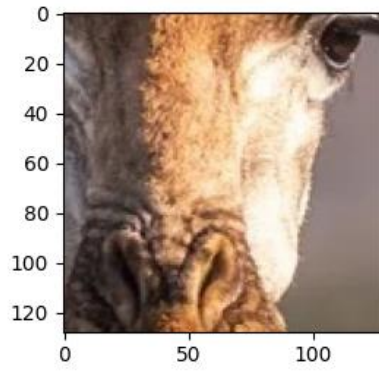
zyr 300 200 4-2-2 100pro



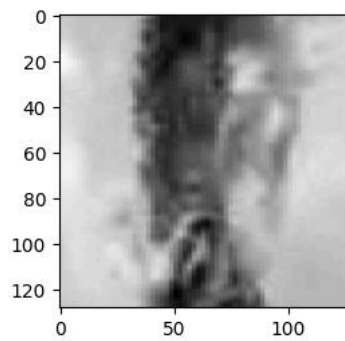
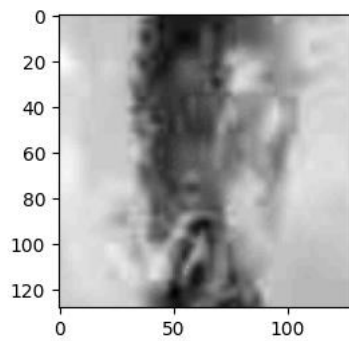
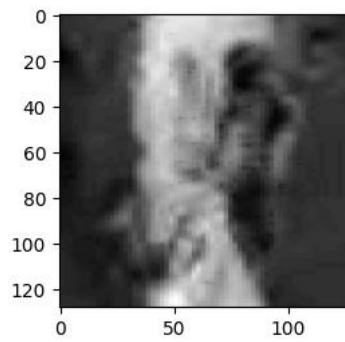
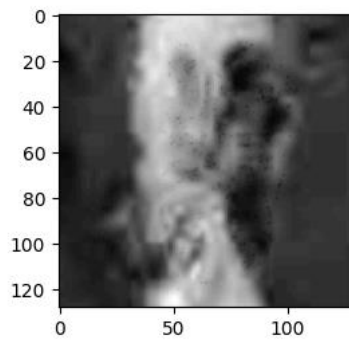
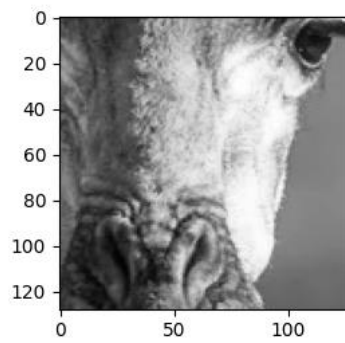
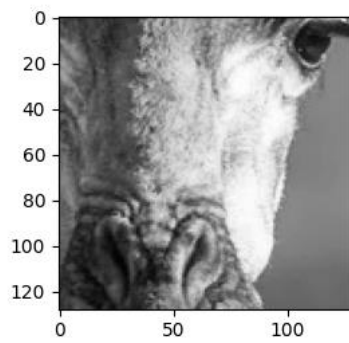
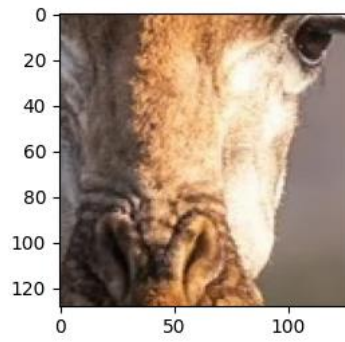
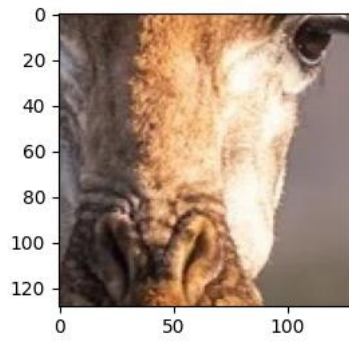
zyr 300 200 4-4-4



zyr 450 300 4-2-2 50pro



zyr 450 300 4-2-2 100pro



zyr 450 300 4-4-4

