FSRCNN 분석

염지현

fsrcnn

```
image get 소요시간:  0.014342546463012695
bicubic 소요시간:  0.09615921974182129
cuda:0
모델 선언 소요시간:  5.661431550979614
weight get 소요시간:  0.052438974380493164
(1080, 1920, 3)
input image rgb 2 ycbcr 소요시간:  0.06374096870422363
(2160, 3840, 3)
bicubc cbcr 추출 소요시간:  0.2928283214569092
first part 소요시간:  0.90513014793396
seconde part 소요시간:  0.2594871520996094
last part 소요시간:  0.016087055206298828
predict 소요시간:  1.1830356121063232
preds shape: (2160, 3840)
preds shape: (2160, 3840, 3)
YCbCr 2 rgb 소요시간:  0.37018656730651855
image 저장 시간:  0.04874110221862793
total 소요 시간:  7.783818244934082
```

Gpu 올리기 전

```
In [15]:
import time
test(2, './fsrcnn_x2/epoch_2412.pth', './data/museum-01.
```

```
image get 소요시간:  0.012927055358886719
bicubic 소요시간:  0.10208368301391602
cuda:0
모델 선언 소요시간:  0.05422401428222656
weight get 소요시간:  0.05247998237609863
(1080, 1920, 3)
input image rgb 2 ycbcr 소요시간:  0.057517290115356445
(2160, 3840, 3)
bicubc cbcr 추출 소요시간:  0.2801058292388916
first part 소요시간:  0.0005011558532714844
seconde part 소요시간:  0.0006990432739257812
last part 소요시간:  0.0002009868621826172
predict 소요시간:  0.0016565322875976562
preds shape: (2160, 3840)
preds shape: (2160, 3840, 3)
YCbCr 2 rgb 소요시간:  0.3731575012207031
image 저장 시간:  0.04368257522583008
total 소요 시간:  0.9787604808807373
```

```
In [ ]:
import argparse
```

Gpu 올린 후

fsrcnn

```
image get 소요시간:  0.0180966854095459
bicubic 소요시간:  0.10996294021606445
cpu
모델 선언 소요시간:  0.002174854278564453
weight get 소요시간:  0.0012607574462890625
(1080, 1920, 3)
input image rgb 2 ycbcr 소요시간:  0.057330846786
(2160, 3840, 3)
bicubc cbcr 추출 소요시간:  0.28966331481933594
first part 소요시간:  0.3614318370819092
seconde part 소요시간:  0.936859130859375
last part 소요시간:  0.6386339664459229
predict 소요시간:  1.9420223236083984
preds shape: (2160, 3840)
preds shape: (2160, 3840, 3)
YCbCr 2 rgb 소요시간:  0.29337525367736816
image 저장 시간:  0.03874635696411133
total 소요 시간:  2.75400447845459
```

CPU

SRCNN

```
71  output.save(args.image_file.replace('.',
72  print("image 저장 소요시간: ", time.time(
73
74  print("SRCNN TOTAL TIME: ", time.time() -
```

```
cuda:0
SRCNN 모델 선언:  6.963290452957153
파라미터 get 소요시간:  0.011447429656982422
image get 소요시간:  0.009117603302001953
bicubic 소요 시간:  0.10237526893615723
rgb 2 ycbcr 소요시간:  0.2478356361389160 2
Conv1 소요시간:  1.366018295288086
Conv2 소요시간:  1.3830626010894775
Conv3 소요시간:  0.7448289394378662
predict 소요시간:  3.495123863220215
ycbcr 2 rgb 소요시간:  0.265625
image 저장 소요시간:  0.15192961692810059
SRCNN TOTAL TIME:  11.24793815612793
```

Gpu 올리기 전

```
73
74  print("SRCNN TOTAL TIME: ", time.time() -
```

```
cuda:0
SRCNN 모델 선언:  0.011836051940917969
파라미터 get 소요시간:  0.011500358581542969
image get 소요시간:  0.006452083587646484
bicubic 소요 시간:  0.09378290176391602
rgb 2 ycbcr 소요시간:  0.22120094299316406
Conv1 소요시간:  0.0004119873046875
Conv2 소요시간:  0.0023727416992187 5
Conv3 소요시간:  7.486343383789062e-05
predict 소요시간:  0.0034415721893310547
ycbcr 2 rgb 소요시간:  0.4582483768463135
image 저장 소요시간:  0.1532001495361328
SRCNN TOTAL TIME:  0.960759162902832
```

Gpu 올린 후

SRCNN

```
72 output.save(args.image_file.replace('.', '_srcnn_x{}.'
73 print("image 저장 소요시간: ", time.time() - Tsaveimg)
74
75 print("SRCNN TOTAL TIME: ", time.time() - s)
```

```
cpu
SRCNN 모델 선언:  0.0013518333435058594
파라미터 get 소요시간:  0.0007927417755126953
image get 소요시간:  0.00472569465637207
bicubic 소요 시간:  0.09722614288330078
rgb 2 ycbcr 소요시간:  0.2098250389099121
Conv1 소요시간:  0.480701208114624
Conv2 소요시간:  1.3445730209350586
Conv3 소요시간:  0.4700915813446045
predict 소요시간:  2.2987563610076904
ycbcr 2 rgb 소요시간:  0.18561267852783203
image 저장 소요시간:  0.1352455615997314
SRCNN TOTAL TIME:  2.9346718788146973
```

In [5]:

```
1 image = pil_image.open(args.image_file).convert('RGB')
2
```

CPU

Python 시간 비교

	FSRCNN		SRCNN	
function	CPU	GPU	CPU	GPU
image loading	0.018	0.0335	0.0112	0.00645
bicubic	0.1099	0.0791	0.1158	0.0937
모델 선언	0.0041	0.0542	0.0031	0.0118
weight loading	0.0012	0.0533	0.0015	0.0115
RGB --〉 YCbCr	0.0573	0.0569	0.2265	0.2212
bicubc CbCr 추출	0.2896	0.2667	0	0
FIRST	0.3614	0.0004	0.4013	0.00041
MID	0.9368	0.0007	0.9599	0.00019
LAST	0.6386	0.0002	0.3261	0.00008
predict(first + mid + last)	1.942	0.00192	1.6919	0.00115
YCbCr --〉 RGB	0.2933	0.3441	0.1944	0.4582
image 저장	0.0387	0.0404	0.1459	0.1532
전체 소요 시간	2.754	0.9342	2.3719	0.9578

Python 시간 비교

```python

class SRCNN(nn.Module):
    def __init__(self, num_channels=1):
        super(SRCNN, self).__init__()
        self.conv1 = nn.Conv2d(num_channels, 64, kernel_size=9, padding=9 // 2)
        self.conv2 = nn.Conv2d(64, 32, kernel_size=5, padding=5 // 2)
        self.conv3 = nn.Conv2d(32, num_channels, kernel_size=5, padding=5 // 2)
        self.relu = nn.ReLU(inplace=True)

    def forward(self, x):
        Tconv1 = time.time()
        x = self.relu(self.conv1(x))
        print("Conv1 소요시간: ", time.time() - Tconv1)
        Tconv2 = time.time()
        x = self.relu(self.conv2(x))
        print("Conv2 소요시간: ", time.time() - Tconv2)
        Tconv3 = time.time()
        x = self.conv3(x)
        print("Conv3 소요시간: ", time.time() - Tconv3)
        return x

```

Python 시간 비교

```python
import math
from torch import nn
import time

class FSRCNN(nn.Module):
    def __init__(self, scale_factor, num_channels=1, d=56, s=12, m=4):
        super(FSRCNN, self).__init__()
        self.first_part = nn.Sequential(
            nn.Conv2d(num_channels, d, kernel_size=5, padding=5//2),
            nn.PReLU(d)
        )
        self.mid_part = [nn.Conv2d(d, s, kernel_size=1), nn.PReLU(s)]
        for _ in range(m):
            self.mid_part.extend([nn.Conv2d(s, s, kernel_size=3, padding=3//2), nn.PReLU(s)])
        self.mid_part.extend([nn.Conv2d(s, d, kernel_size=1), nn.PReLU(d)])
        self.mid_part = nn.Sequential(*self.mid_part)
        self.last_part = nn.ConvTranspose2d(d, num_channels, kernel_size=9, stride=scale_factor, padding=9//2,
                                            output_padding=scale_factor-1)

        self._initialize_weights()


    def _initialize_weights(self):
        for m in self.first_part:
            if isinstance(m, nn.Conv2d):
                nn.init.normal_(m.weight.data, mean=0.0, std=math.sqrt(2/(m.out_channels*m.weight.data[0][0].numel())))
                nn.init.zeros_(m.bias.data)
        for m in self.mid_part:
            if isinstance(m, nn.Conv2d):
                nn.init.normal_(m.weight.data, mean=0.0, std=math.sqrt(2/(m.out_channels*m.weight.data[0][0].numel())))
                nn.init.zeros_(m.bias.data)
        nn.init.normal_(self.last_part.weight.data, mean=0.0, std=0.001)
        nn.init.zeros_(self.last_part.bias.data)

    def forward(self, x):
        Tf = time.time()
        x = self.first_part(x)
        print("first part 소요시간: ", time.time() - Tf)
        Tm = time.time()
        x = self.mid_part(x)
        print("seconde part 소요시간: ", time.time() - Tm)
        Tl = time.time()
        x = self.last_part(x)
        print("last part 소요시간: ", time.time() - Tl)
        return x

```