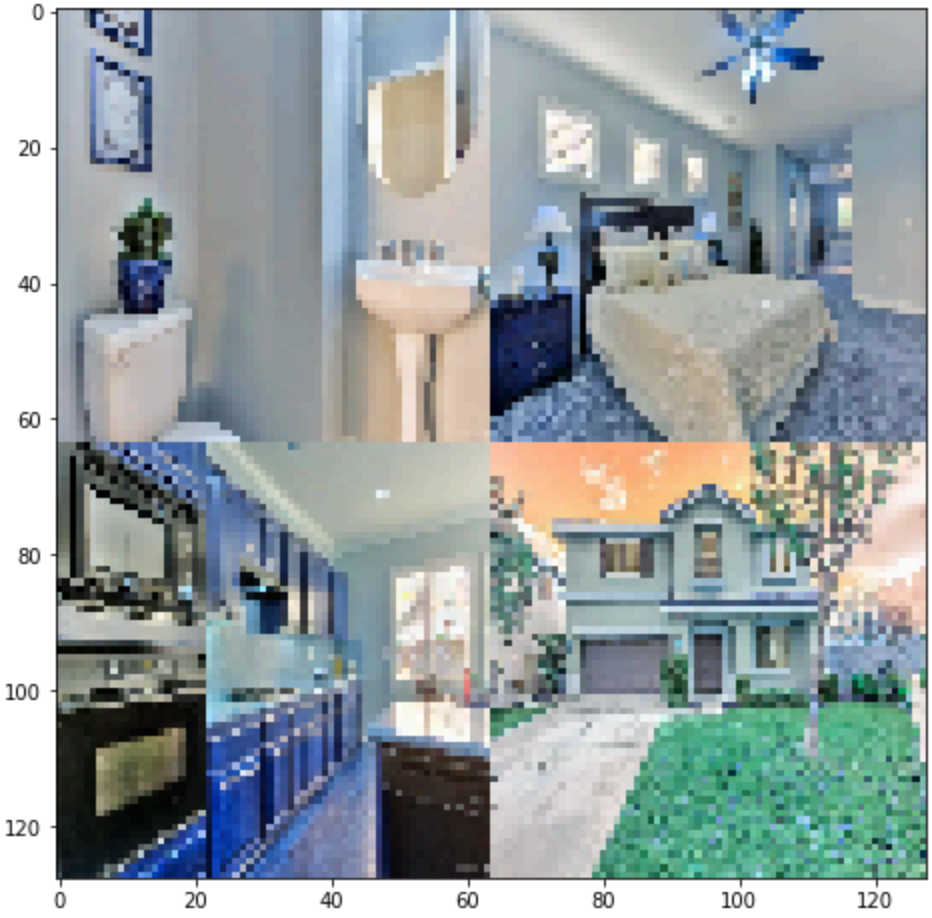
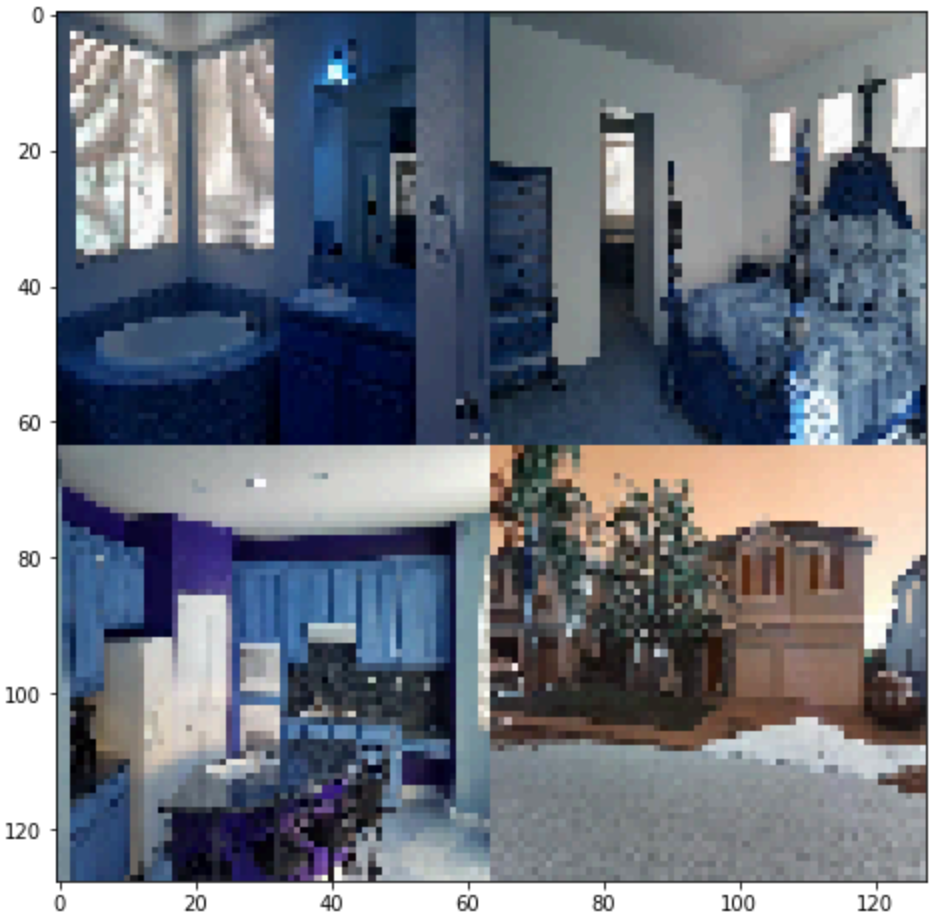

0
20
40
60
80
100
120
0
20
40
60
80
100
120


0
20
40
60
80
100
120
0
20
40
60
80
100
120


0
20
40
60
80
100
120
0
20
40
60
80
100
120

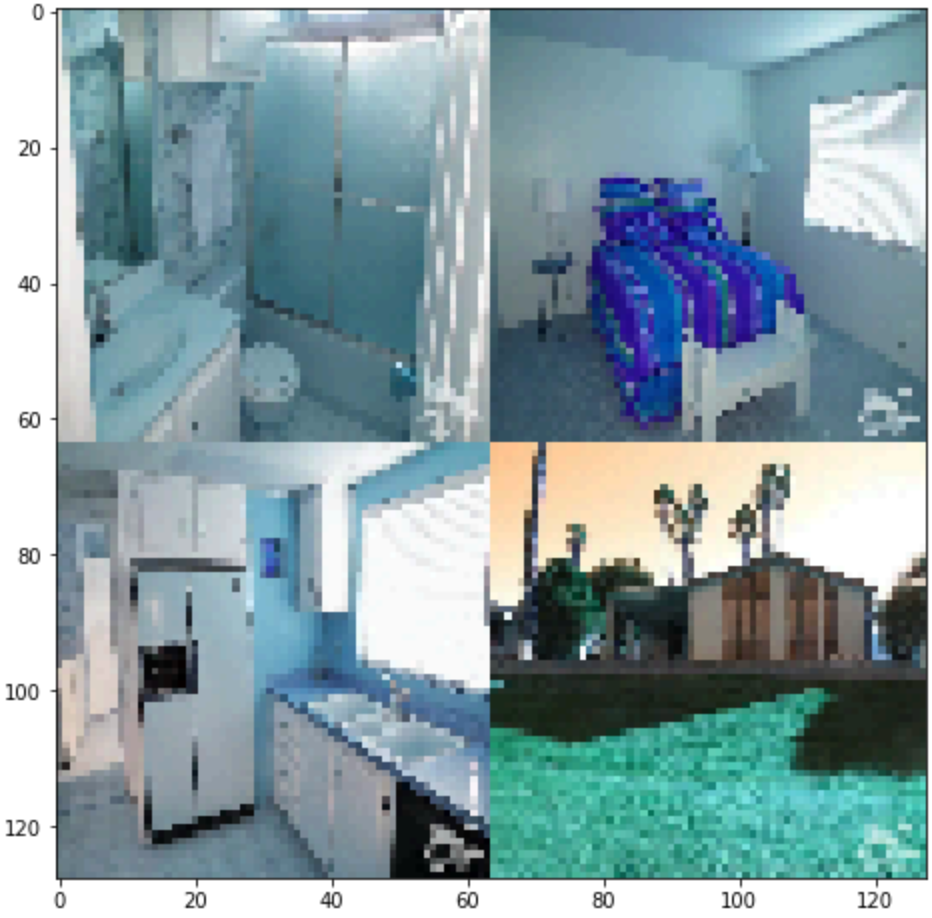

0
20
40
60
80
100
120
0
20
40
60
80
100
120


0
20
40
60
80
100
120
0
20
40
60
80
100
120


0
20
40
60
80
100
120
0
20
40
60
80
100
120


0
20
40
60
80
100
120
0
20
40
60
80
100
120


0
20
40
60
80
100
120
0
20
40
60
80
100
120


0
20
40
60
80
100
120
0
20
40
60
80
100
120


0
20
40
60
80
100
120
0
20
40
60
80
100
120

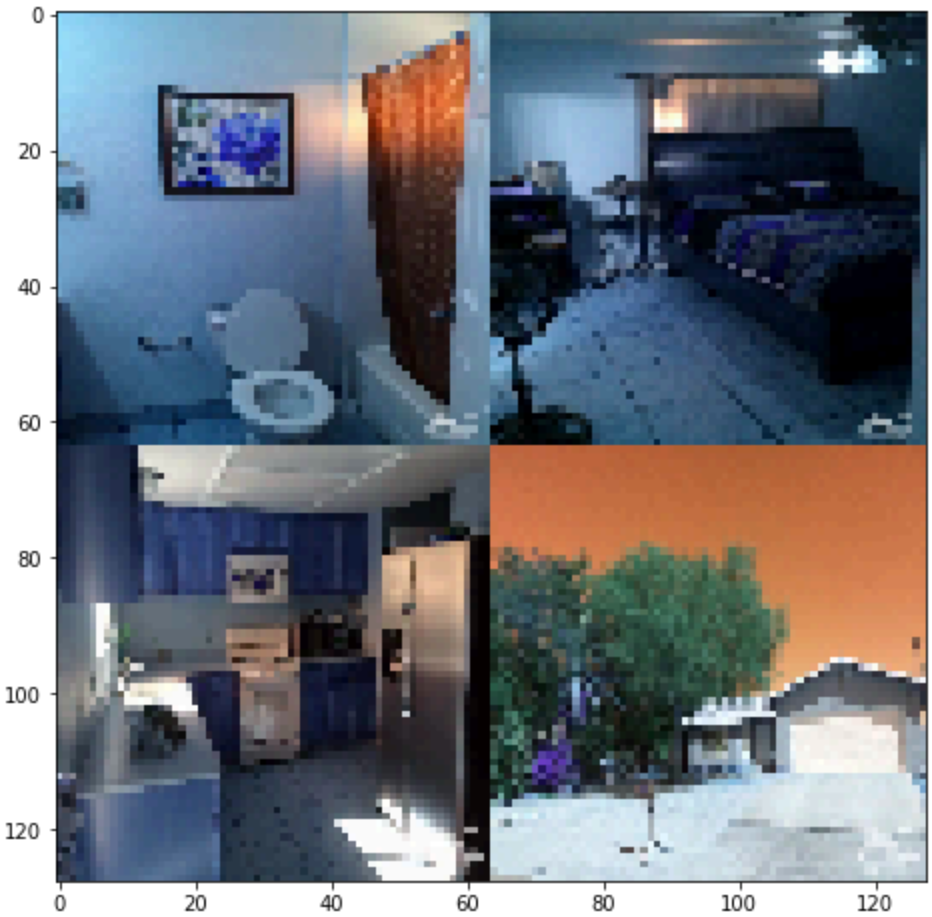

0
20
40
60
80
100
120
0
20
40
60
80
100
120

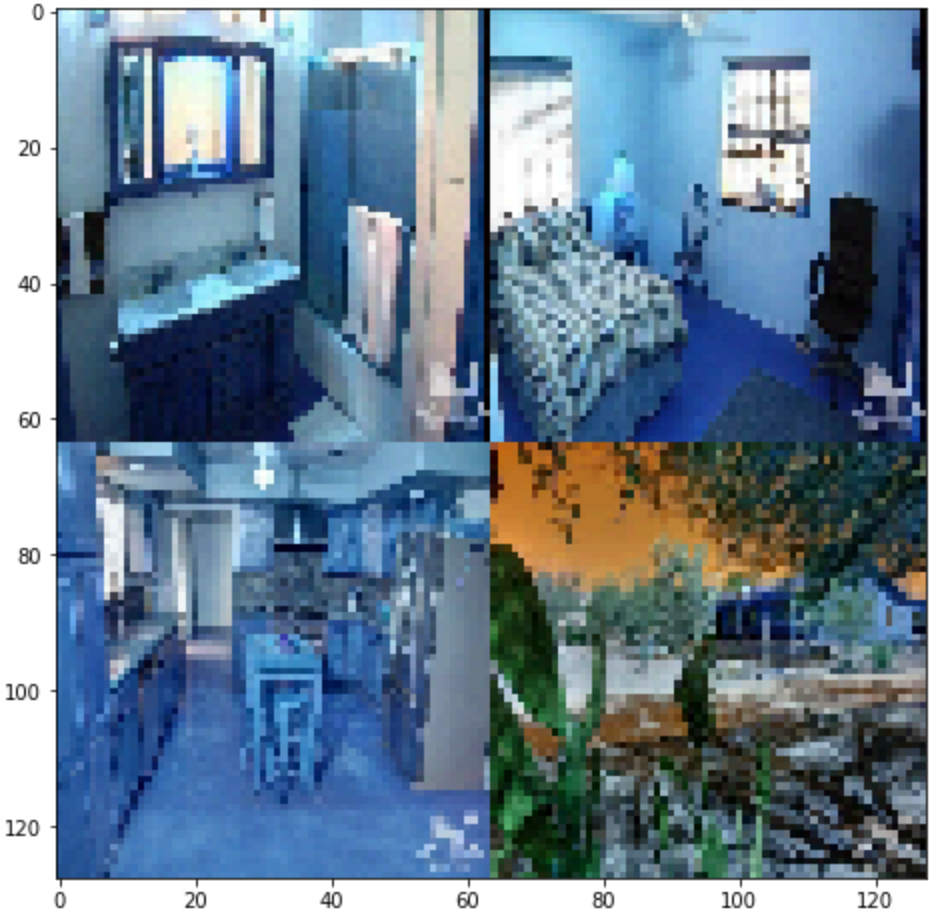

0
20
40
60
80
100
120
0
20
40
60
80
100
120

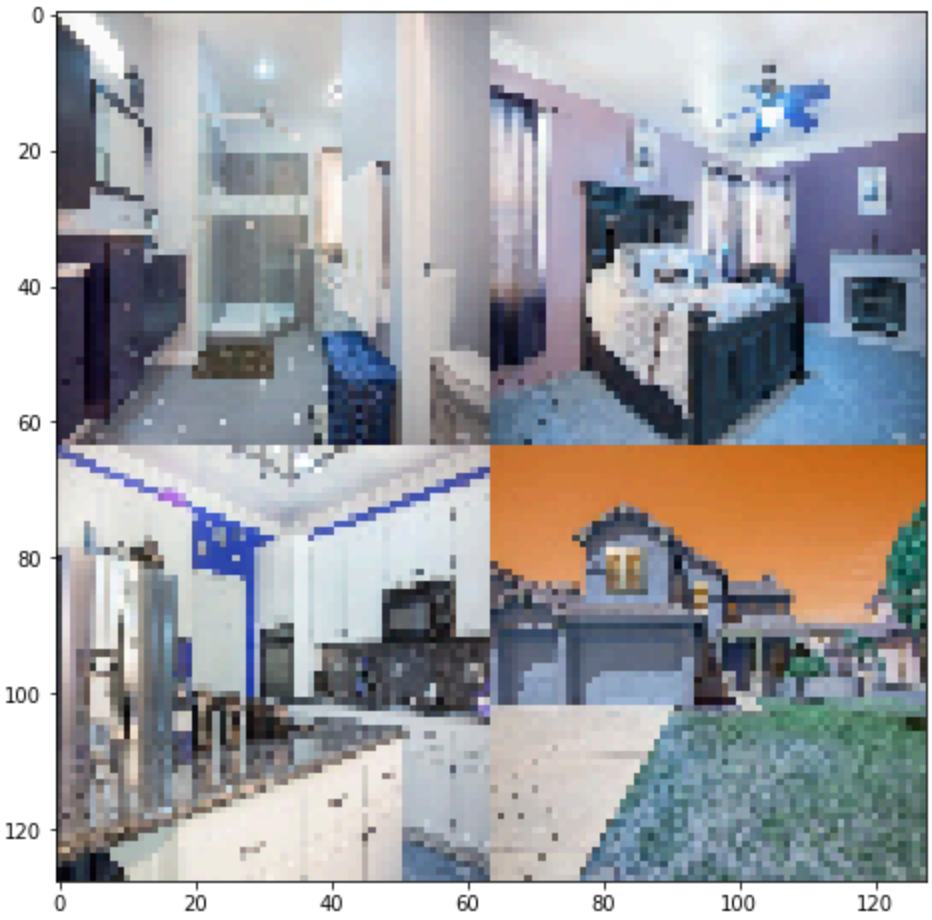

0
20
40
60
80
100
120
0
20
40
60
80
100
120

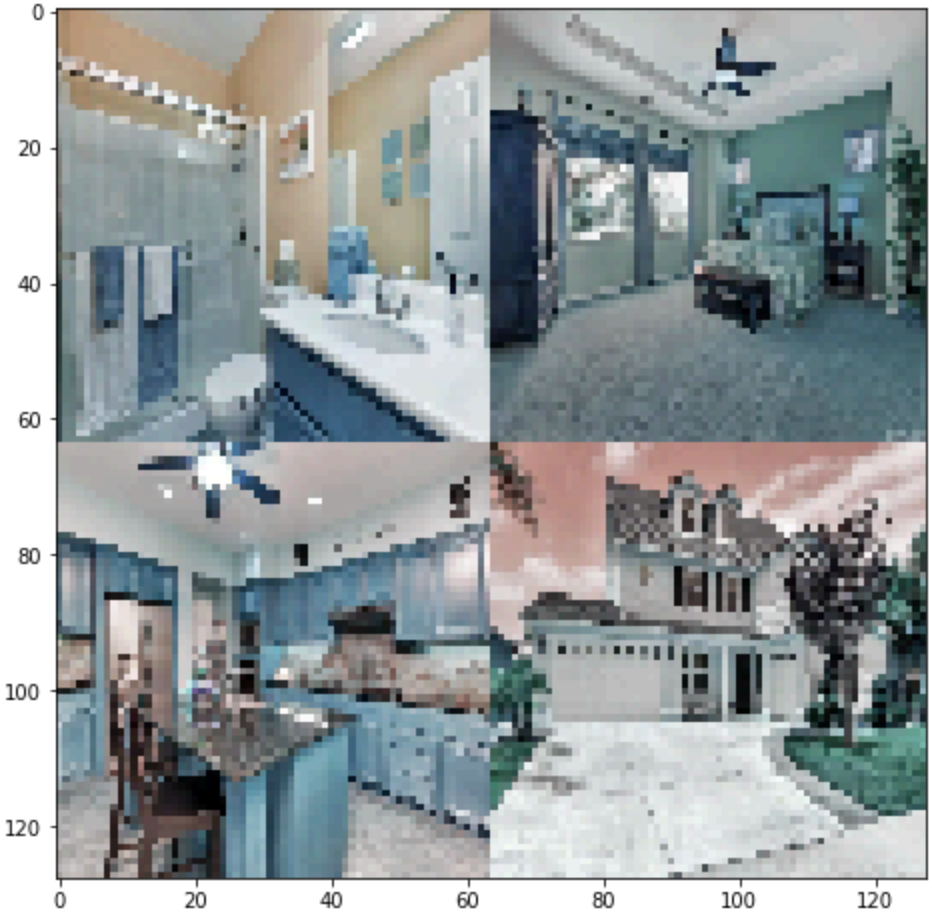

0
20
40
60
80
100
120
0
20
40
60
80
100
120


0
20
40
60
80
100
120
0
20
40
60
80
100
120

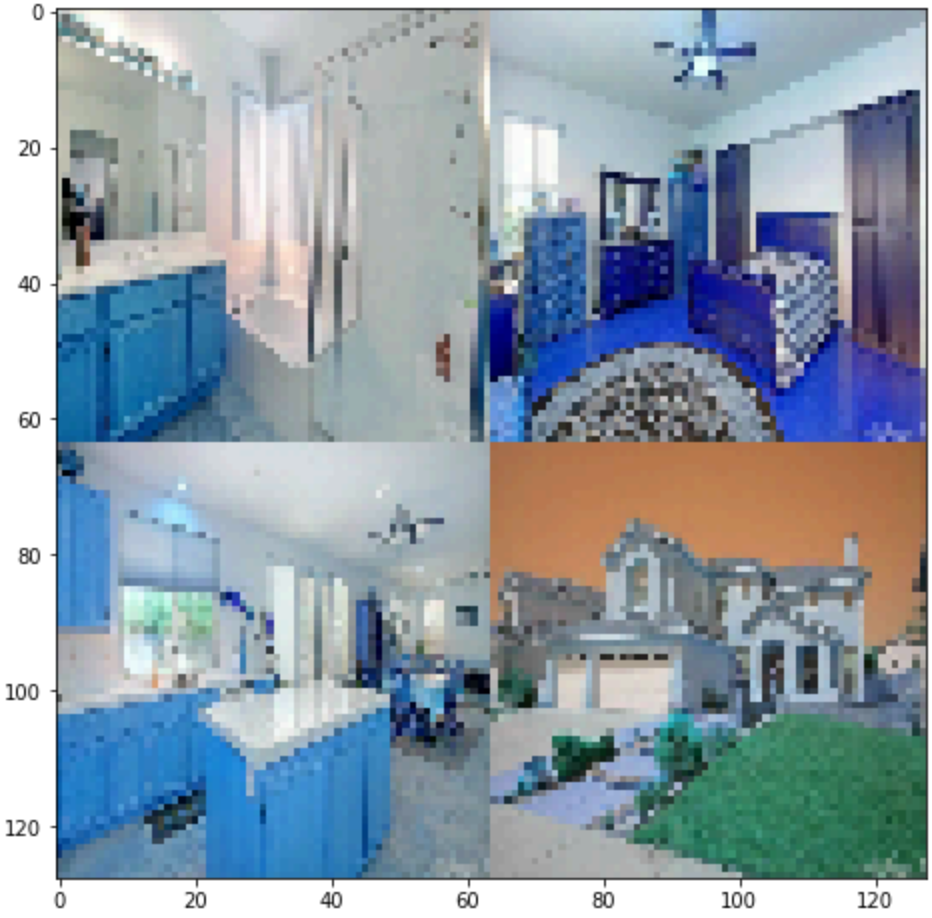

0
20
40
60
80
100
120
0
20
40
60
80
100
120

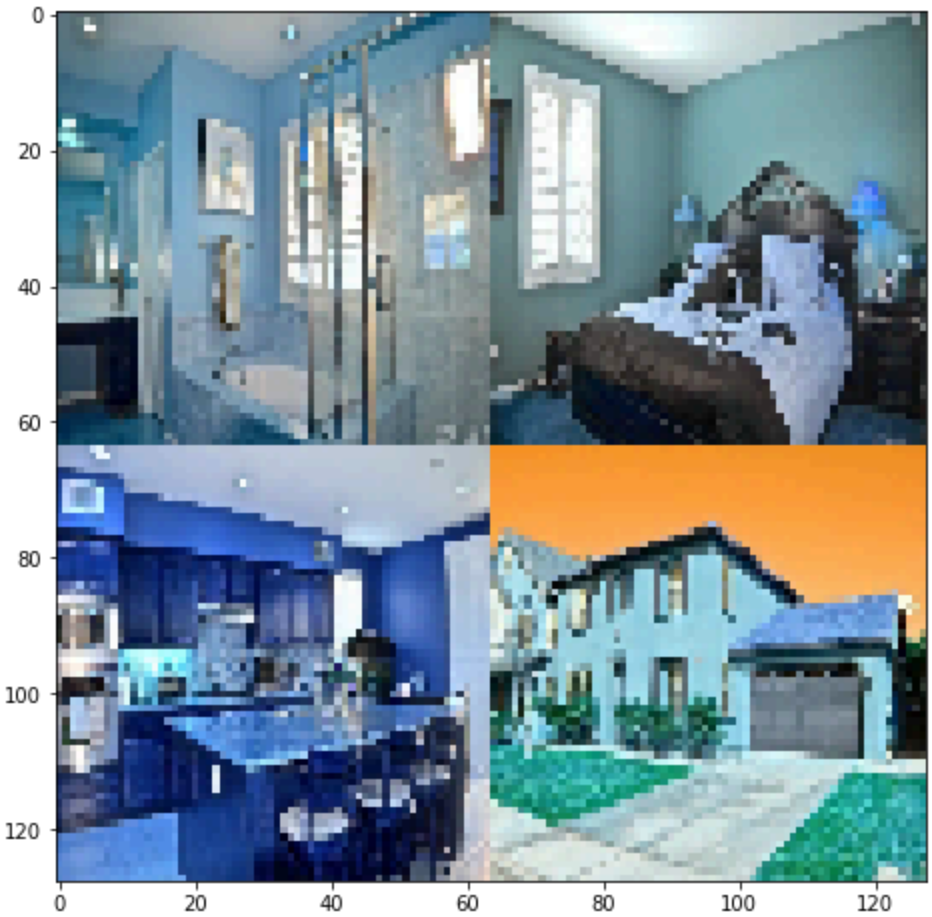

0
20
40
60
80
100
120
0
20
40
60
80
100
120


0
20
40
60
80
100
120
0
20
40
60
80
100
120


0
20
40
60
80
100
120
0
20
40
60
80
100
120

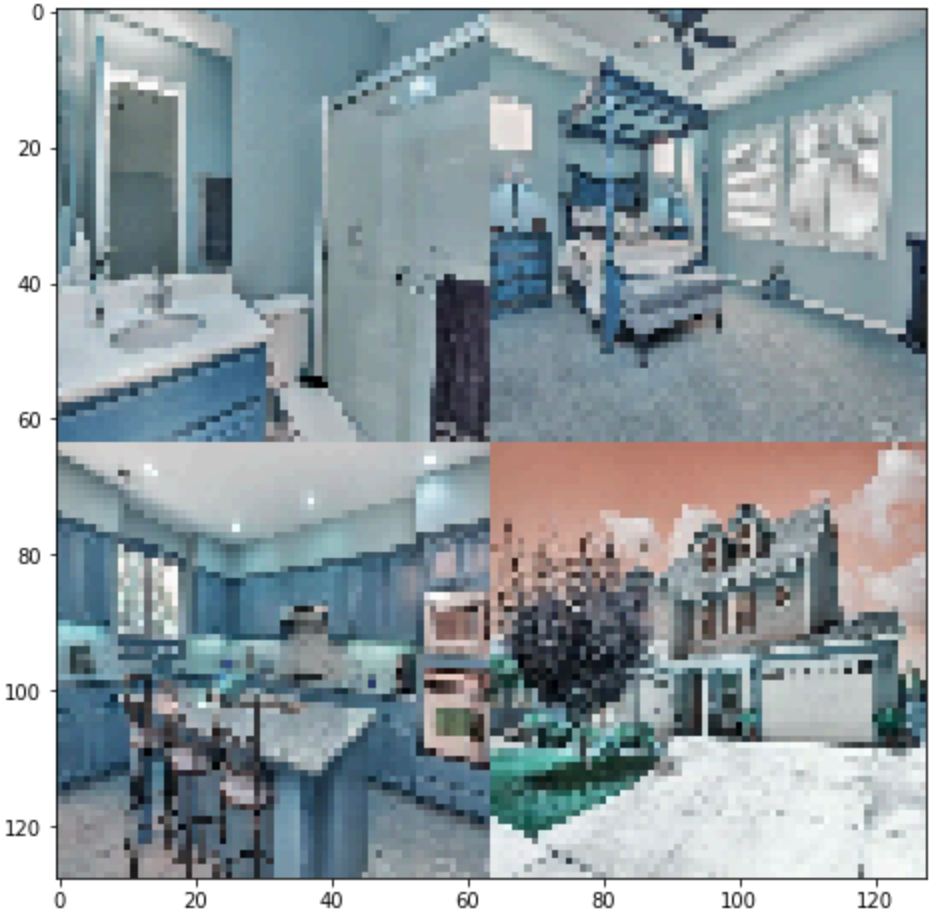

0
20
40
60
80
100
120
0
20
40
60
80
100
120

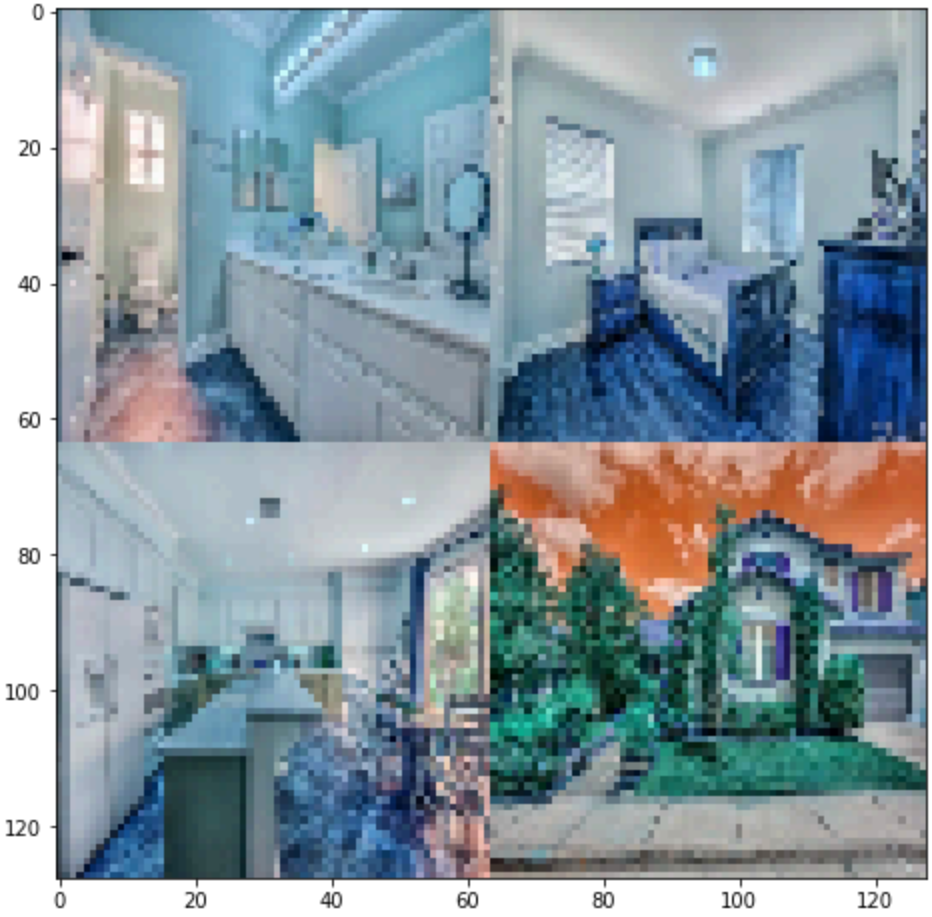

0
20
40
60
80
100
120
0
20
40
60
80
100
120


0
20
40
60
80
100
120
0
20
40
60
80
100
120

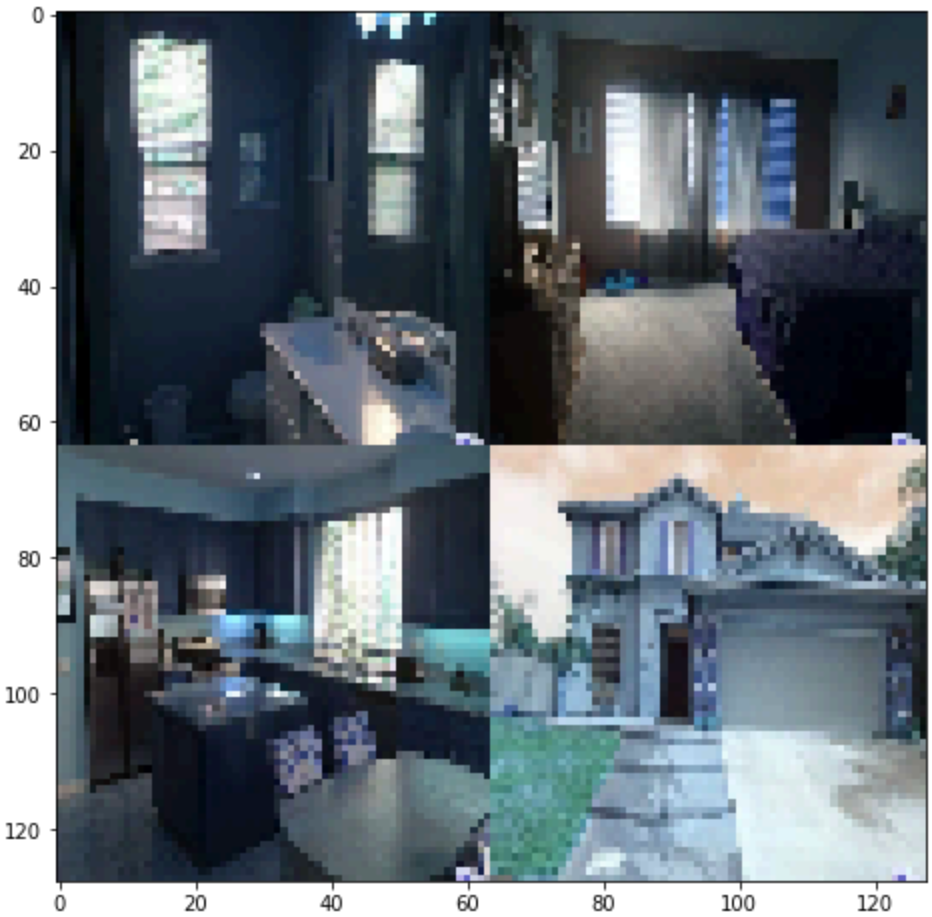

0
20
40
60
80
100
120
0
20
40
60
80
100
120

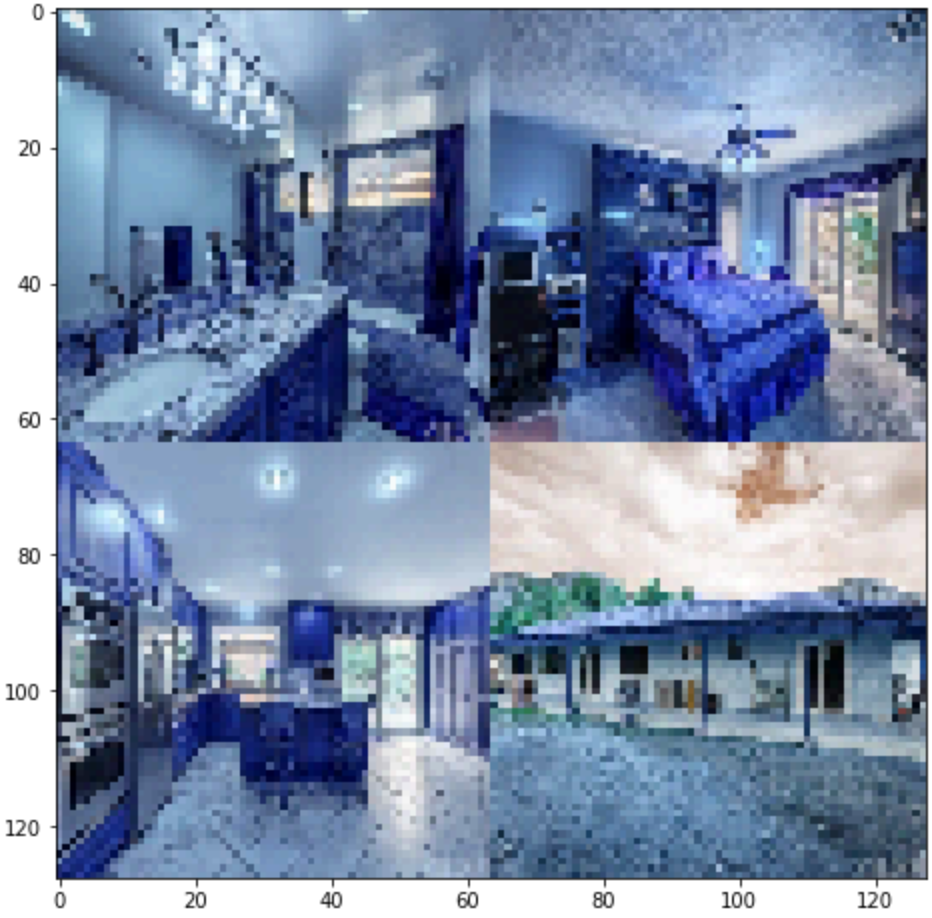

0
20
40
60
80
100
120
0
20
40
60
80
100
120


0
20
40
60
80
100
120
0
20
40
60
80
100
120

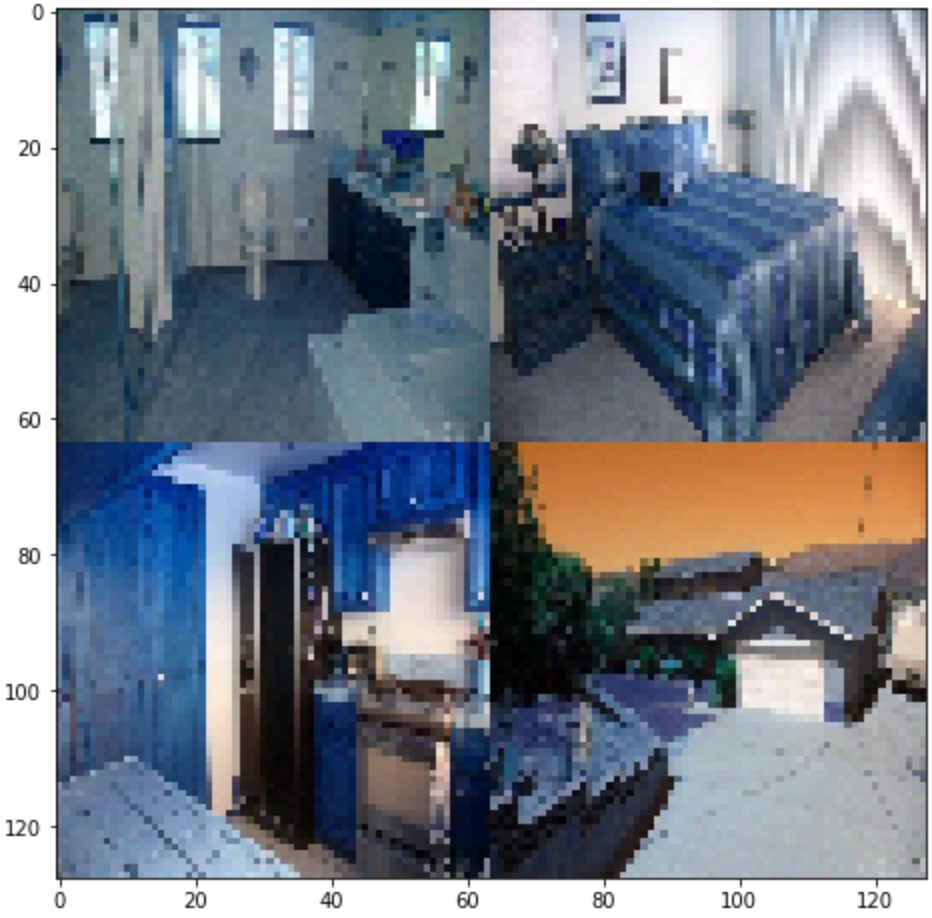

0
20
40
60
80
100
120
0
20
40
60
80
100
120

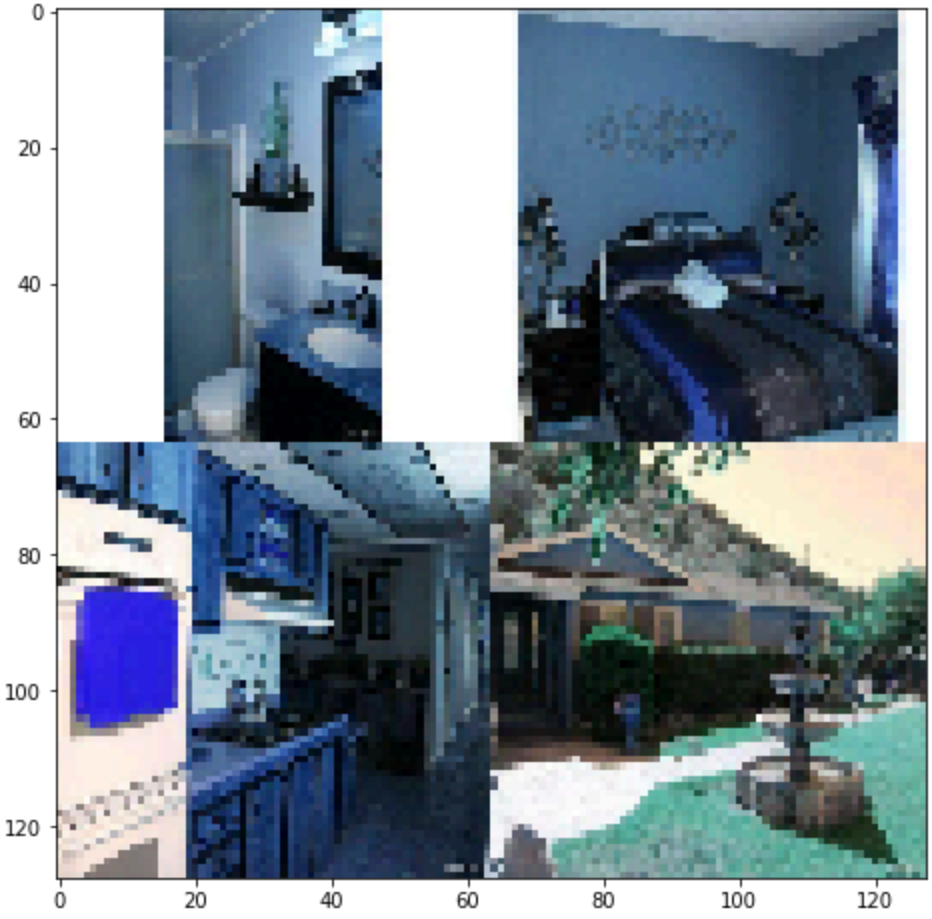

0
20
40
60
80
100
120
0
20
40
60
80
100
120

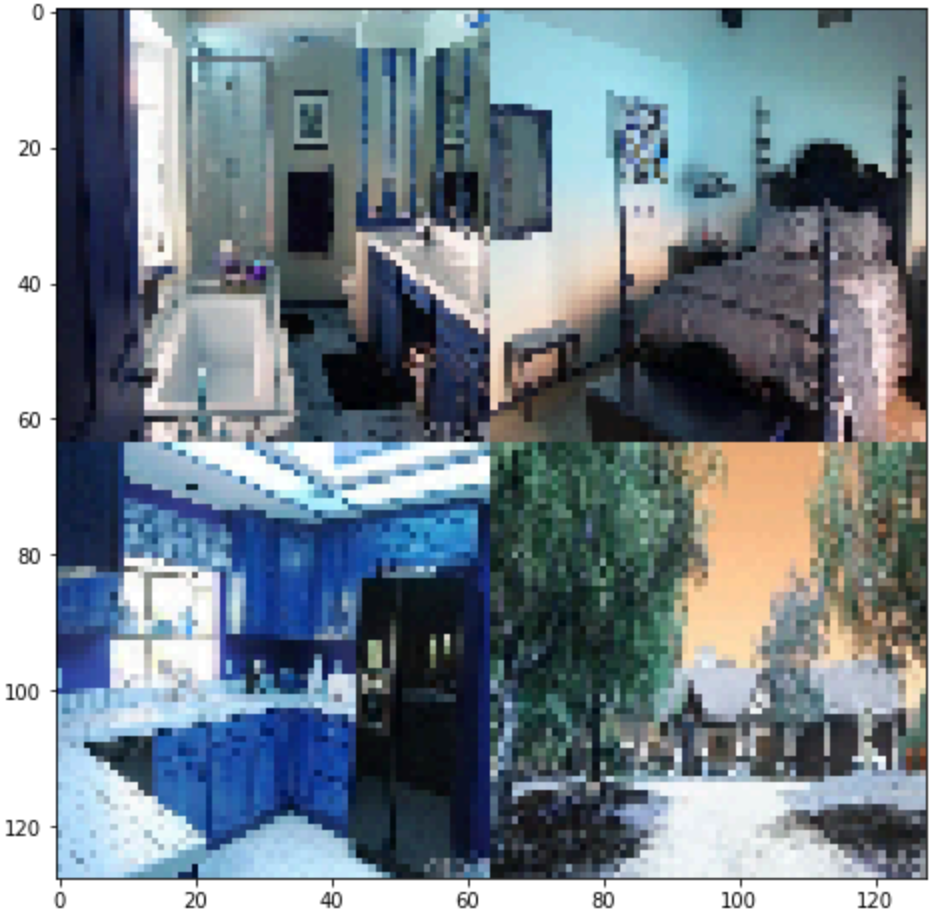

0
20
40
60
80
100
120
0
20
40
60
80
100
120

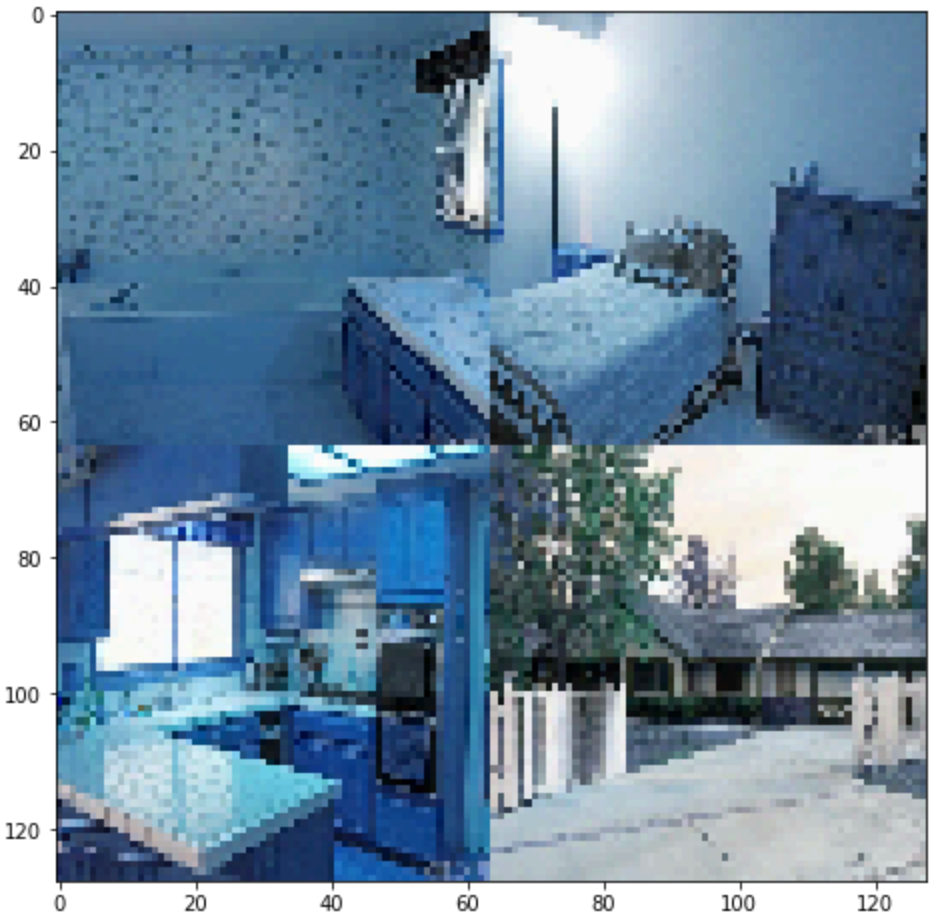

0
20
40
60
80
100
120
0
20
40
60
80
100
120


0
20
40
60
80
100
120
0
20
40
60
80
100
120

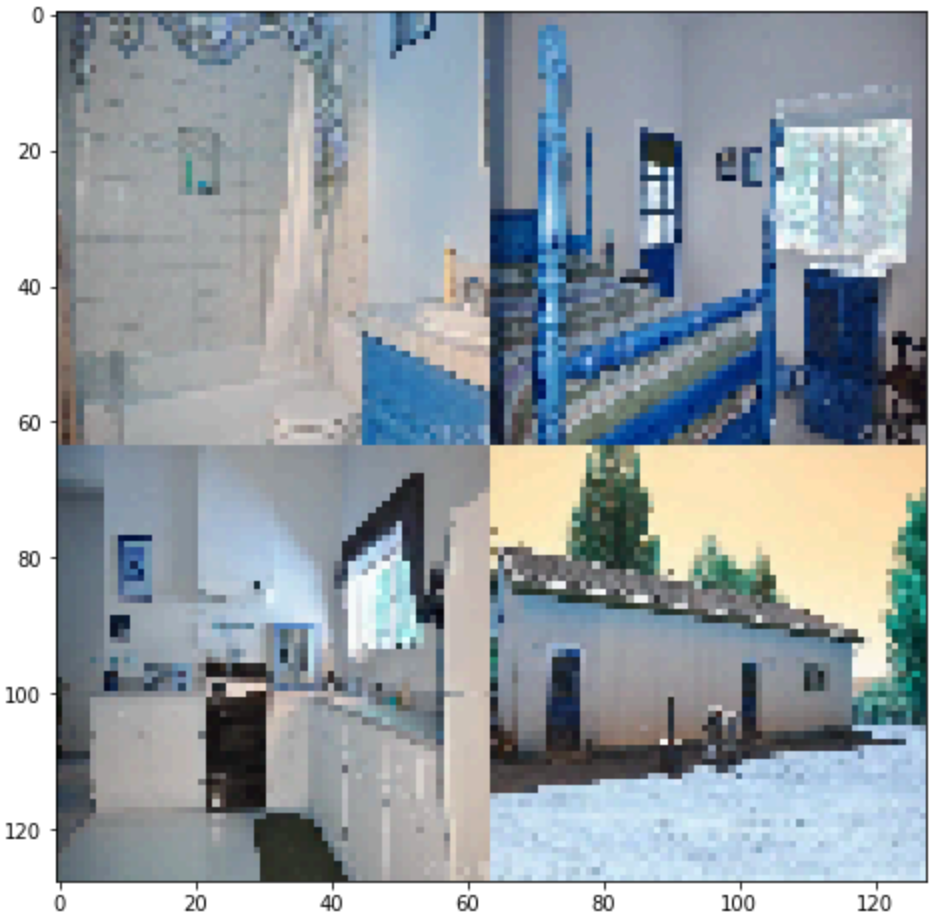

0
20
40
60
80
100
120
0
20
40
60
80
100
120


0
20
40
60
80
100
120
0
20
40
60
80
100
120


0
20
40
60
80
100
120
0
20
40
60
80
100
120


0
20
40
60
80
100
120
0
20
40
60
80
100
120


0
20
40
60
80
100
120
0
20
40
60
80
100
120


0
20
40
60
80
100
120
0
20
40
60
80
100
120


0
20
40
60
80
100
120
0
20
40
60
80
100
120


0
20
40
60
80
100
120
0
20
40
60
80
100
120


0
20
40
60
80
100
120
0
20
40
60
80
100
120


0
20
40
60
80
100
120
0
20
40
60
80
100
120

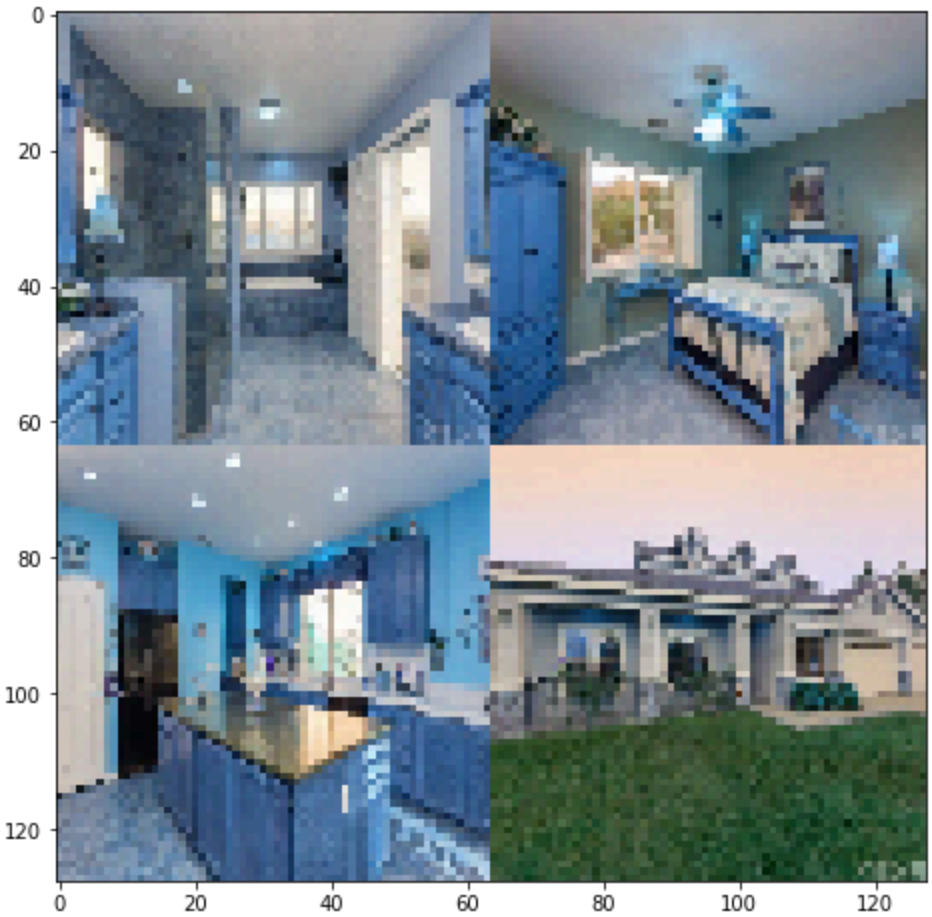

0
20
40
60
80
100
120
0
20
40
60
80
100
120

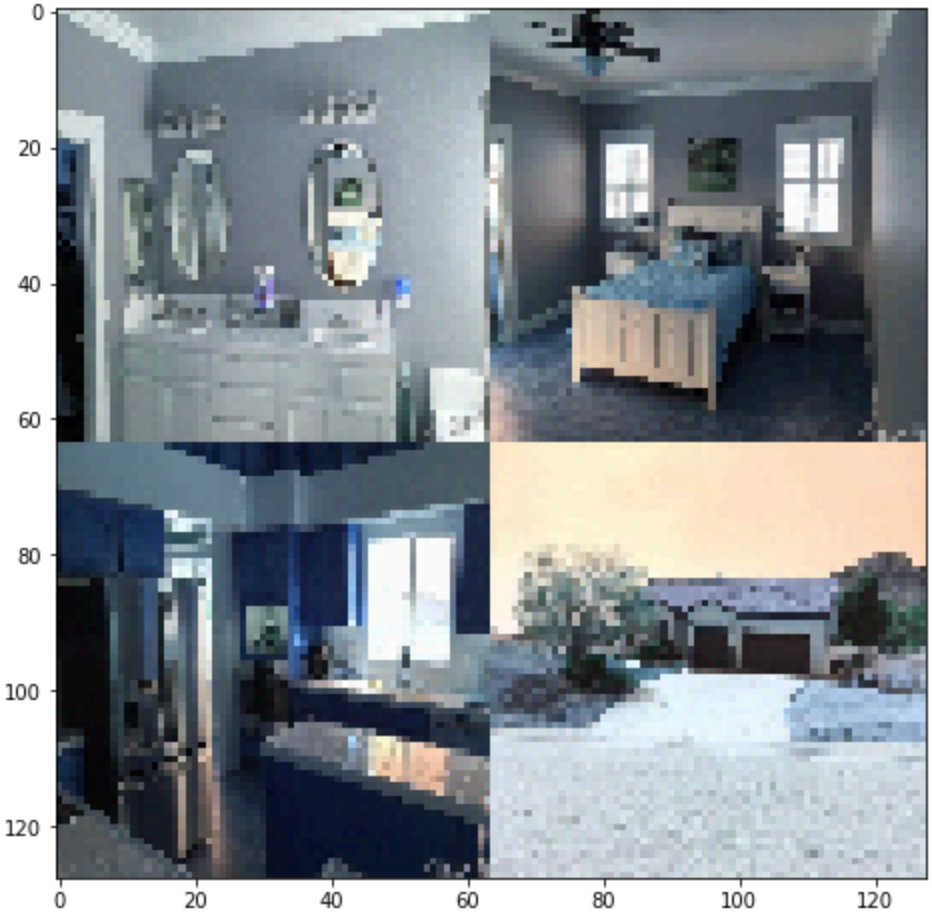

0
20
40
60
80
100
120
0
20
40
60
80
100
120


0
20
40
60
80
100
120
0
20
40
60
80
100
120

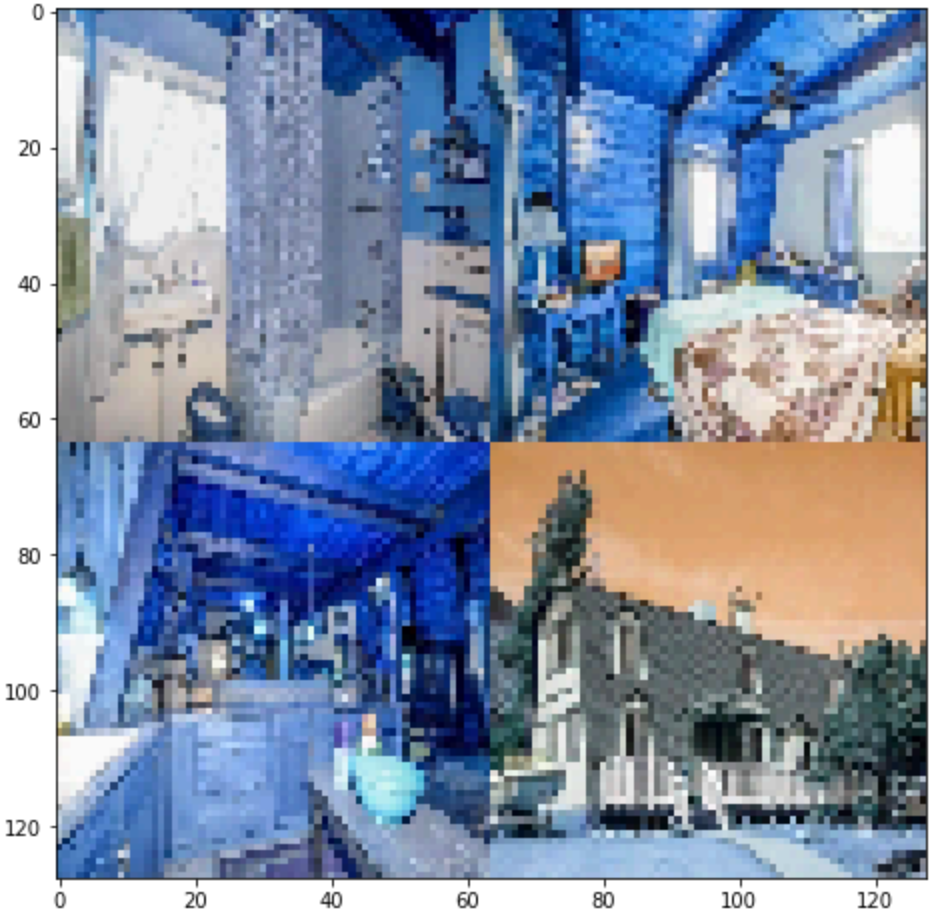

0
20
40
60
80
100
120
0
20
40
60
80
100
120


0
20
40
60
80
100
120
0
20
40
60
80
100
120

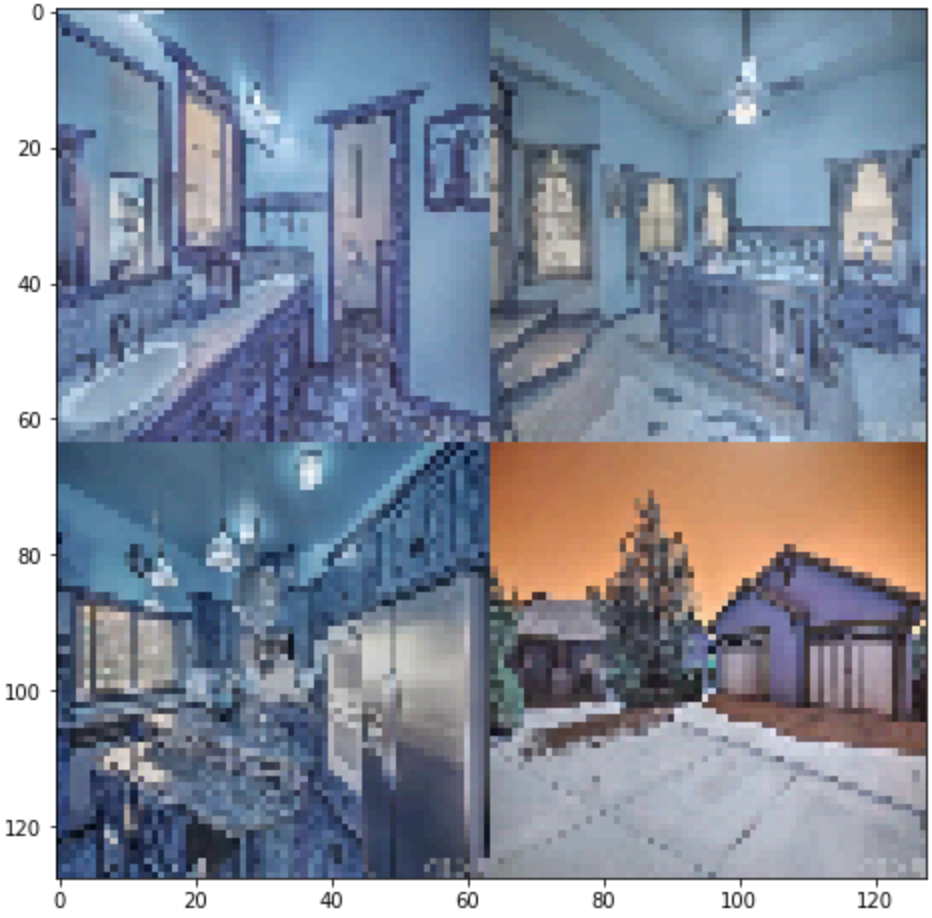

0
20
40
60
80
100
120
0
20
40
60
80
100
120

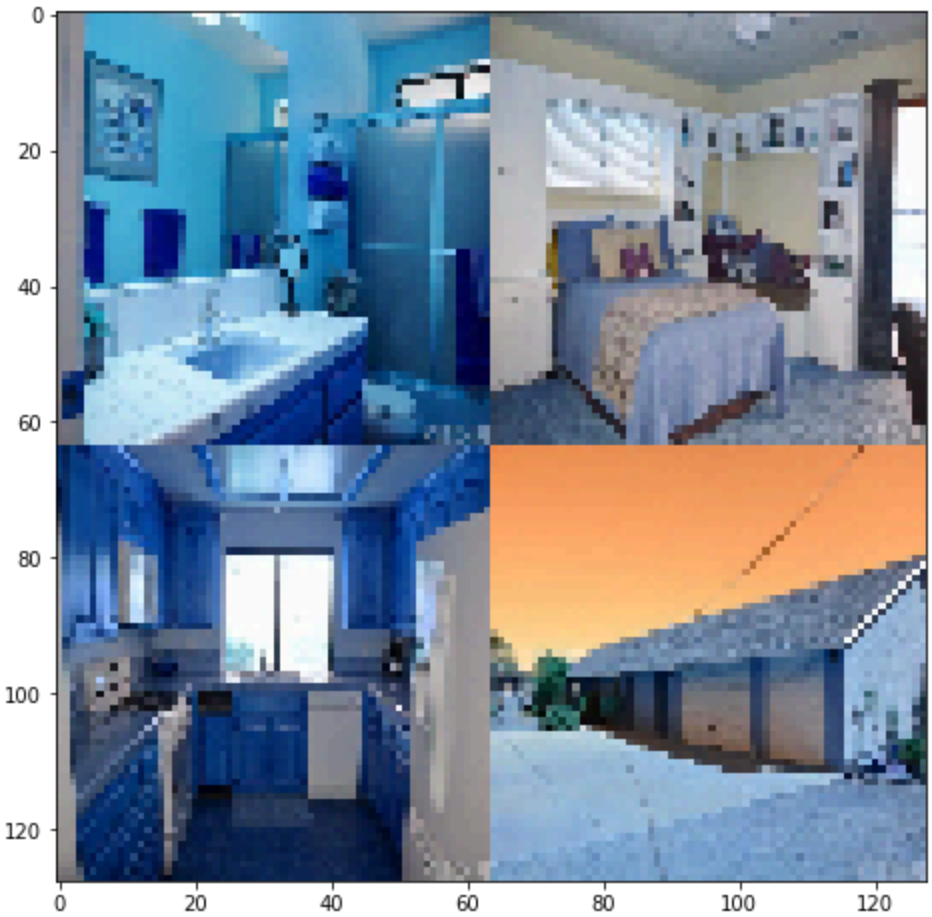

0
20
40
60
80
100
120
0
20
40
60
80
100
120


0
20
40
60
80
100
120
0
20
40
60
80
100
120

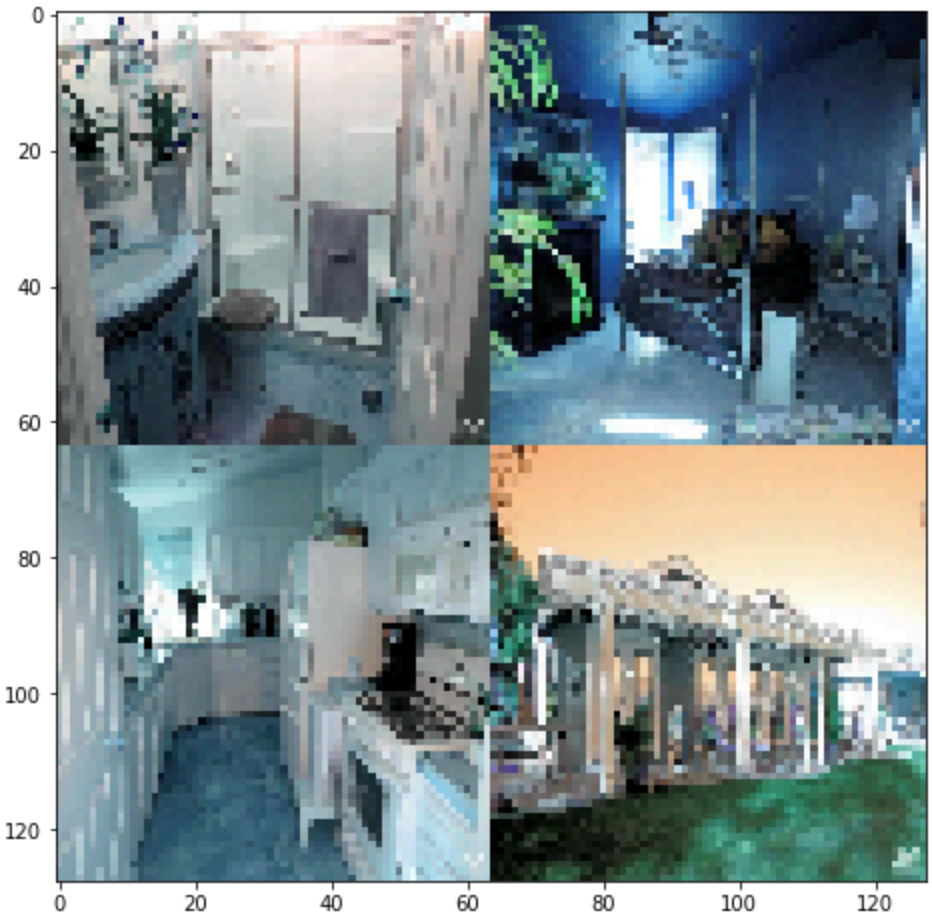

0
20
40
60
80
100
120
0
20
40
60
80
100
120

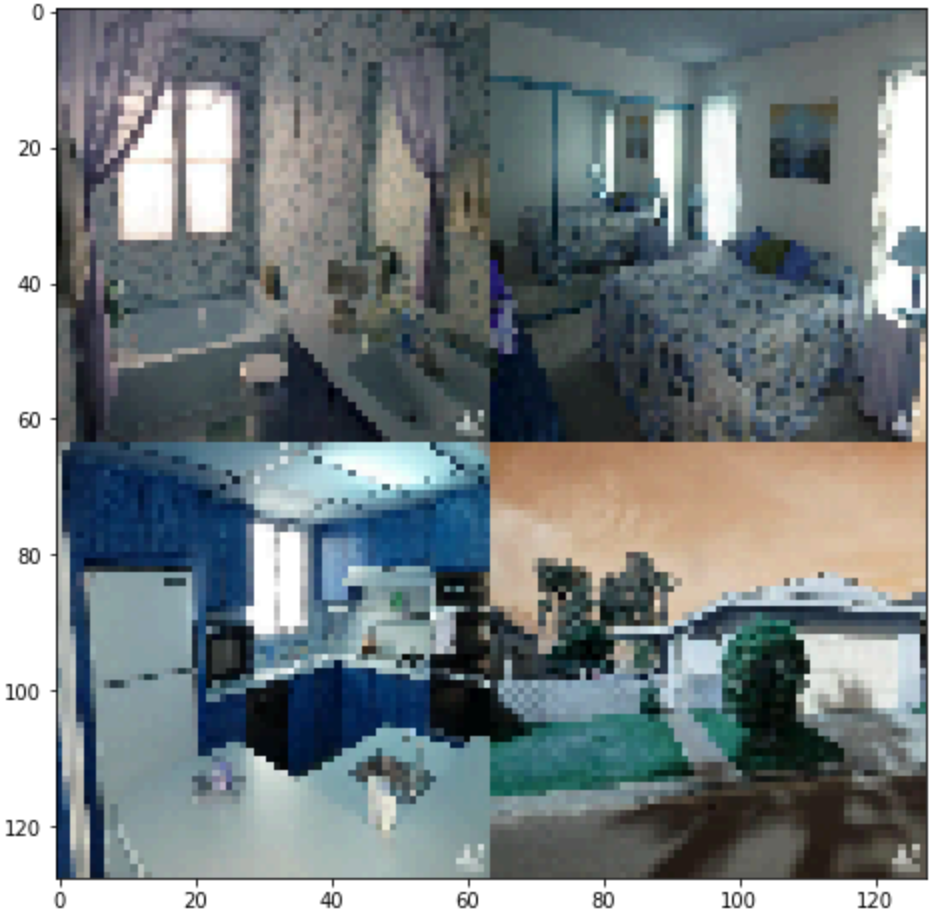

0
20
40
60
80
100
120
0
20
40
60
80
100
120


0
20
40
60
80
100
120
0
20
40
60
80
100
120

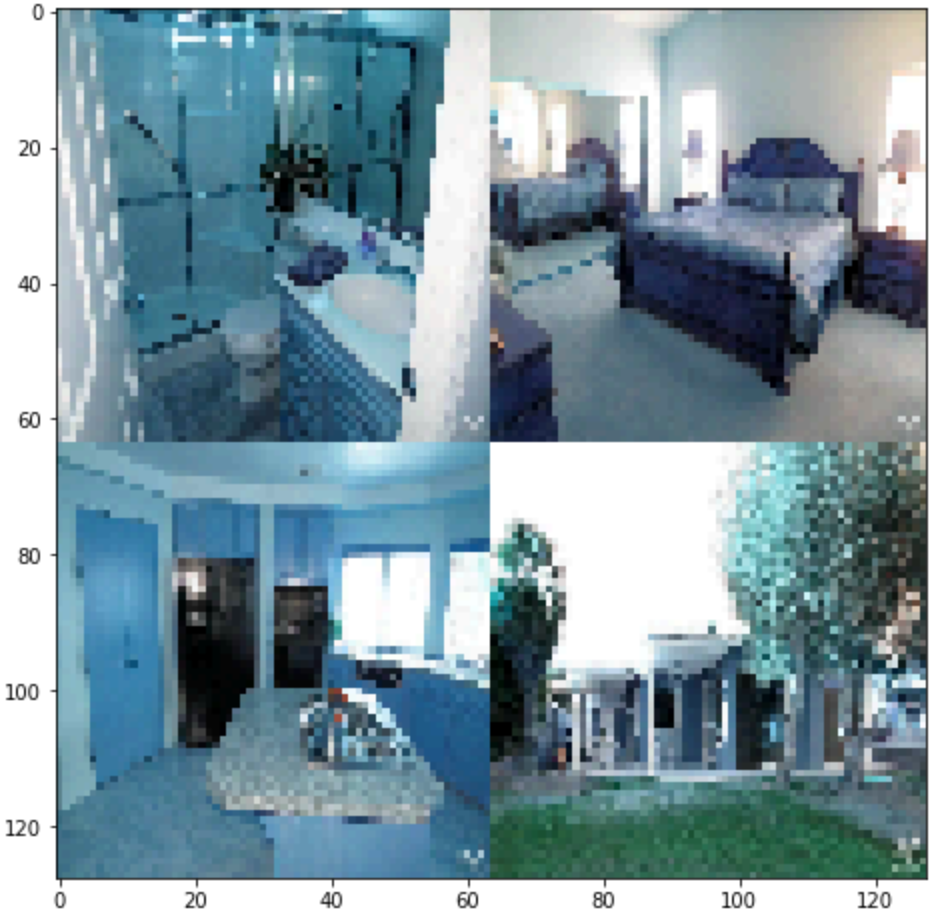

0
20
40
60
80
100
120
0
20
40
60
80
100
120


0
20
40
60
80
100
120
0
20
40
60
80
100
120

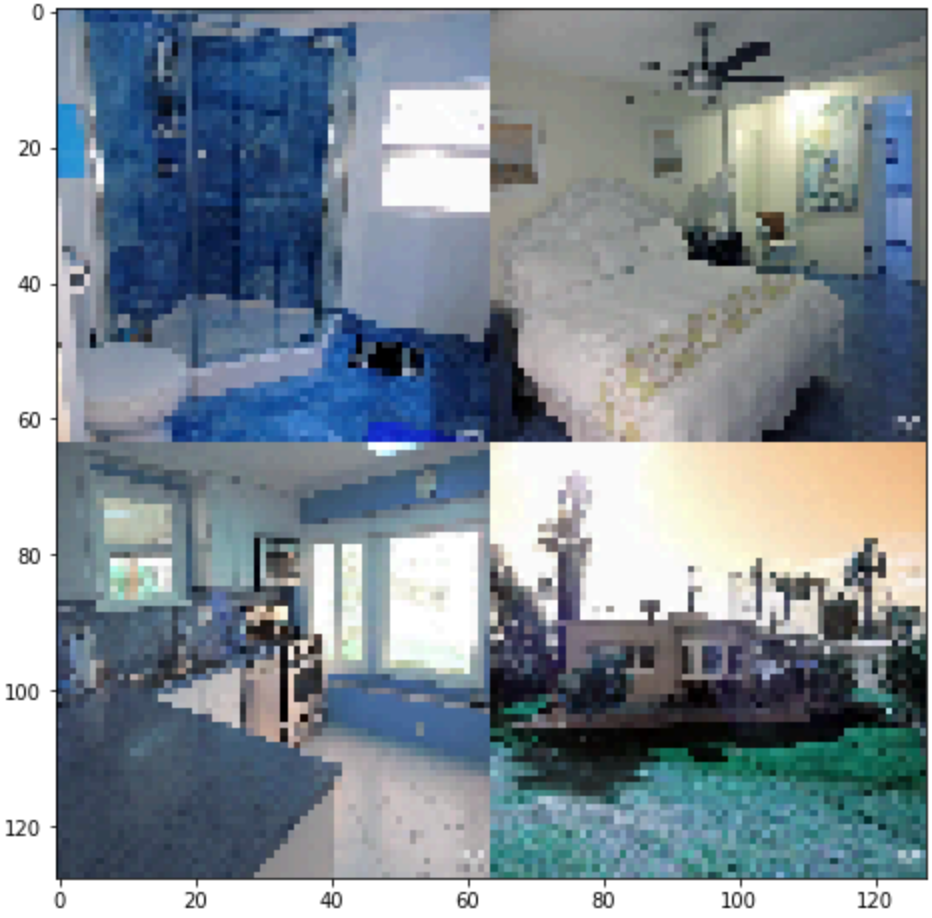

0
20
40
60
80
100
120
0
20
40
60
80
100
120


0
20
40
60
80
100
120
0
20
40
60
80
100
120


0
20
40
60
80
100
120
0
20
40
60
80
100
120


0
20
40
60
80
100
120
0
20
40
60
80
100
120

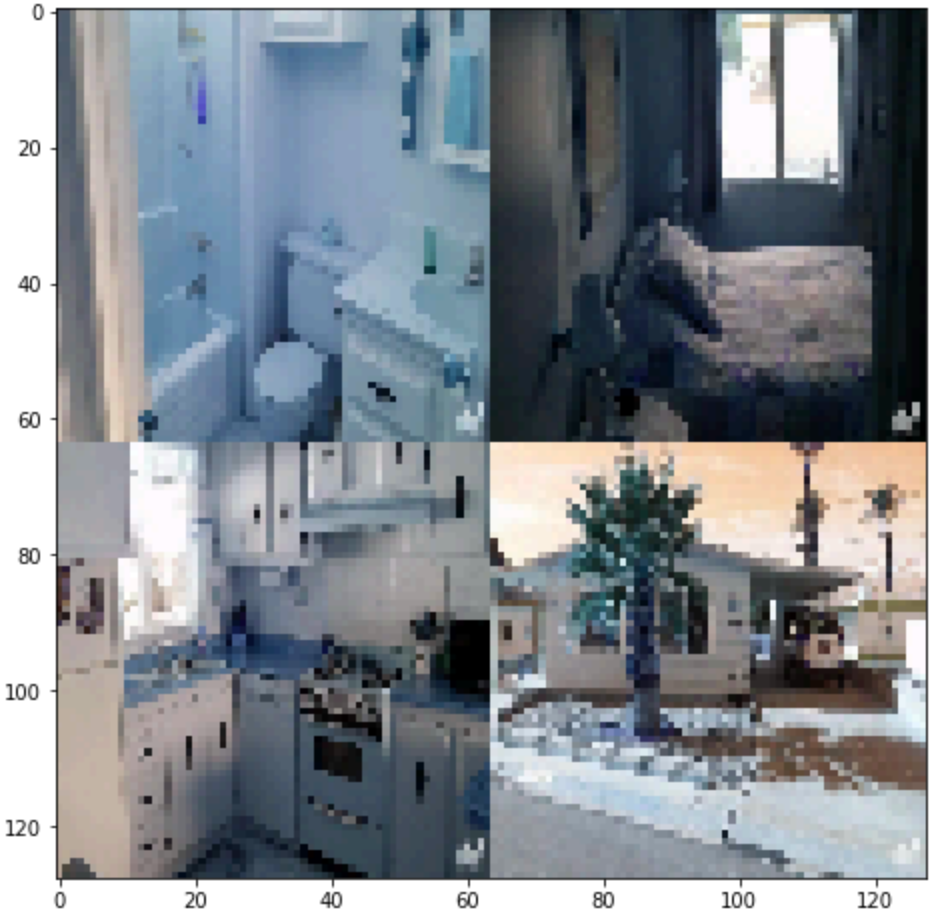

0
20
40
60
80
100
120
0
20
40
60
80
100
120


0
20
40
60
80
100
120
0
20
40
60
80
100
120


0
20
40
60
80
100
120
0
20
40
60
80
100
120


0
20
40
60
80
100
120
0
20
40
60
80
100
120


0
20
40
60
80
100
120
0
20
40
60
80
100
120

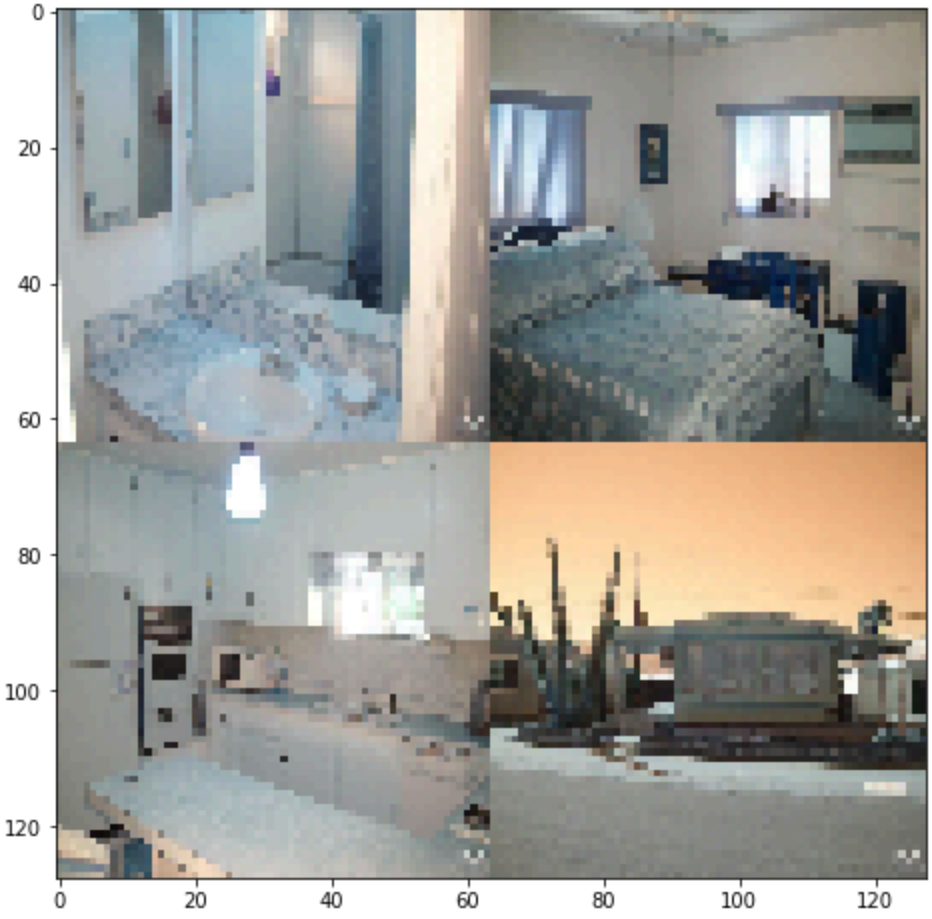

0
20
40
60
80
100
120
0
20
40
60
80
100
120


0
20
40
60
80
100
120
0
20
40
60
80
100
120


0
20
40
60
80
100
120
0
20
40
60
80
100
120

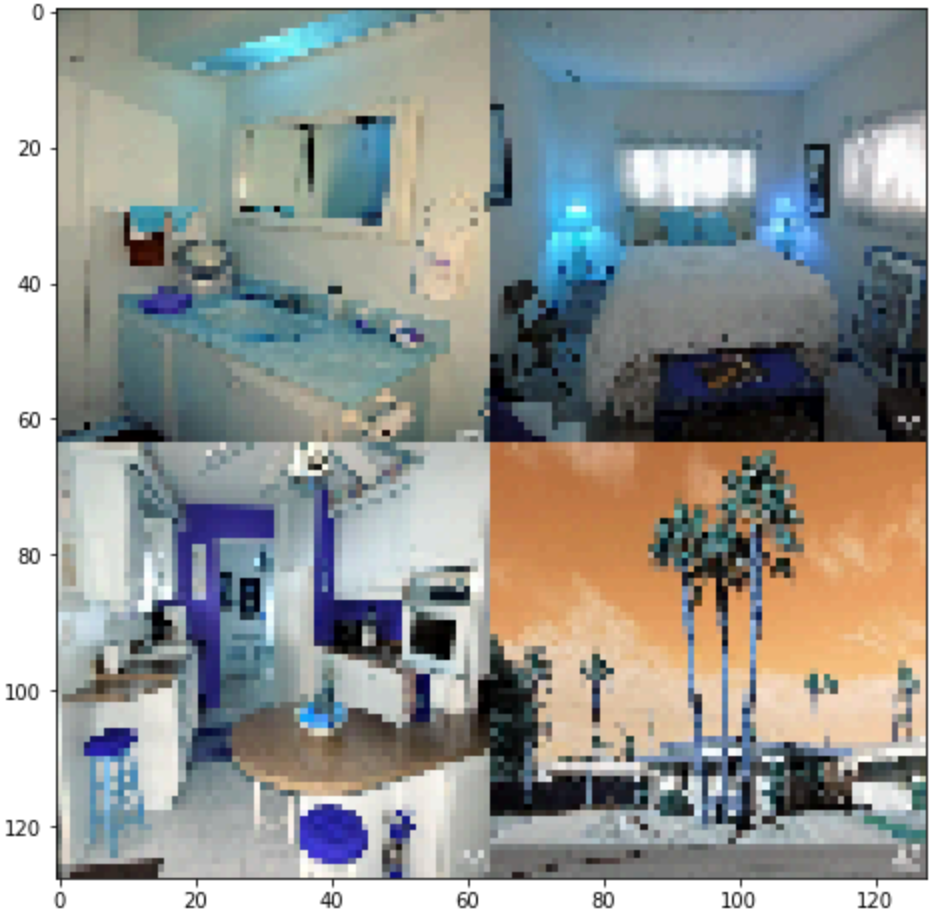

0
20
40
60
80
100
120
0
20
40
60
80
100
120

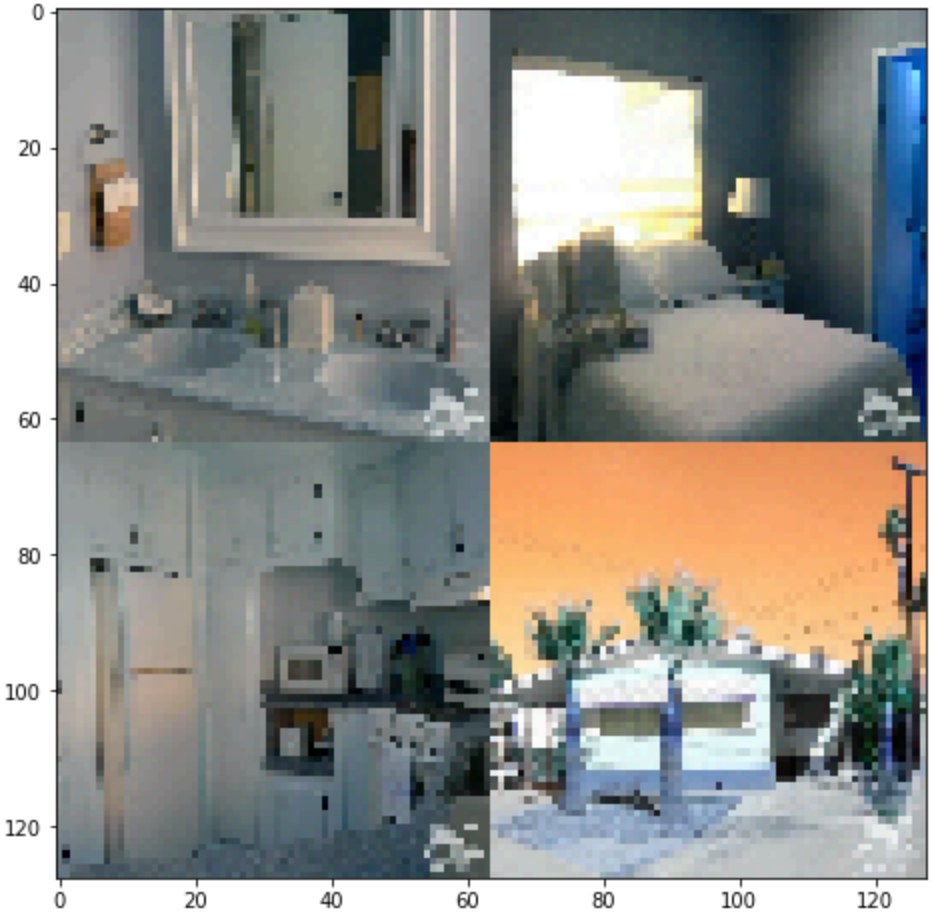

0
20
40
60
80
100
120
0
20
40
60
80
100
120

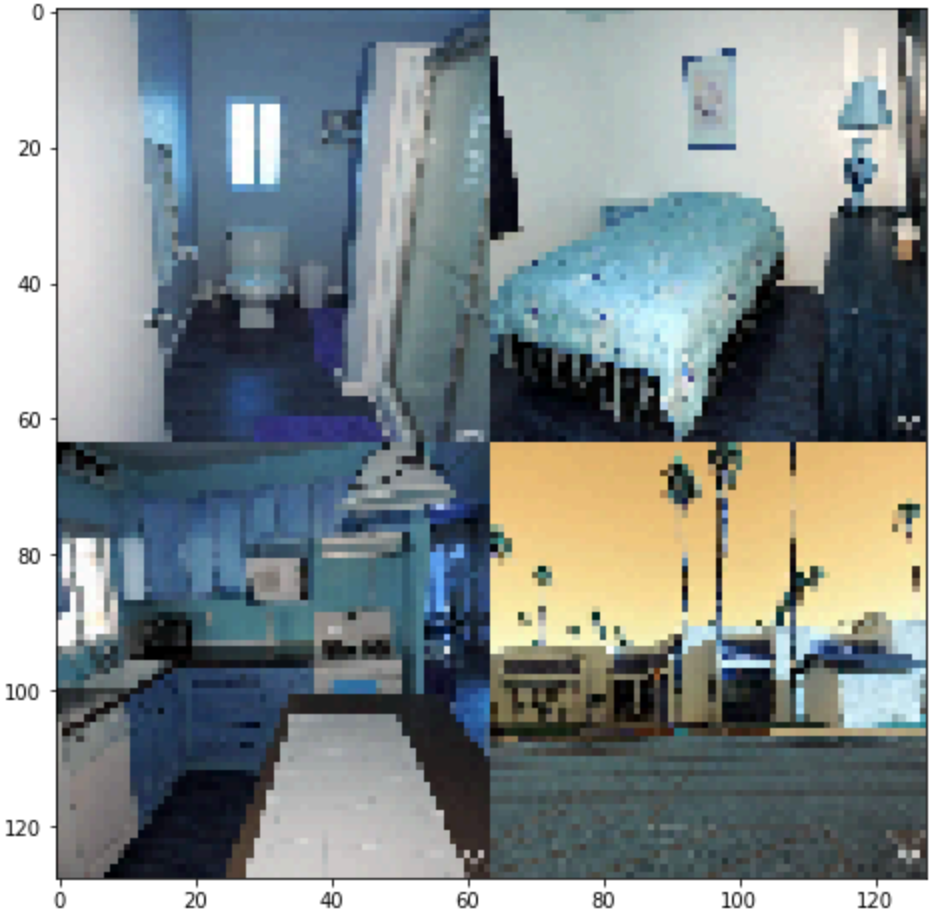

0
20
40
60
80
100
120
0
20
40
60
80
100
120


0
20
40
60
80
100
120
0
20
40
60
80
100
120

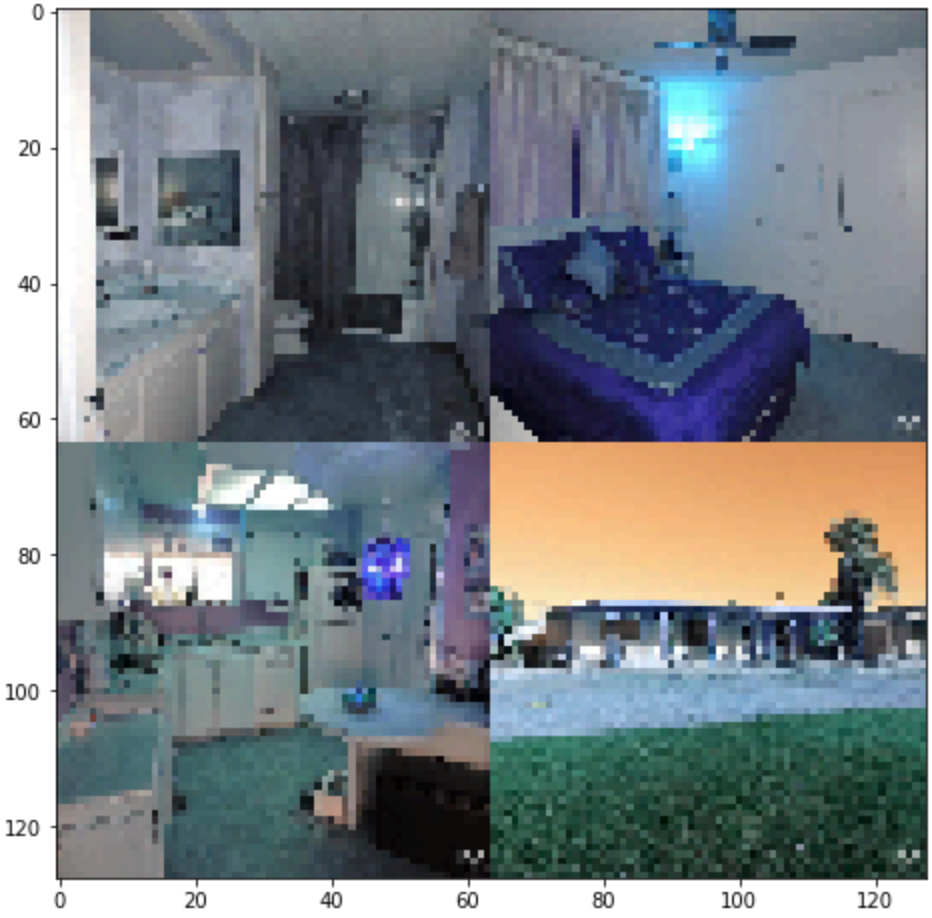

0
20
40
60
80
100
120
0
20
40
60
80
100
120

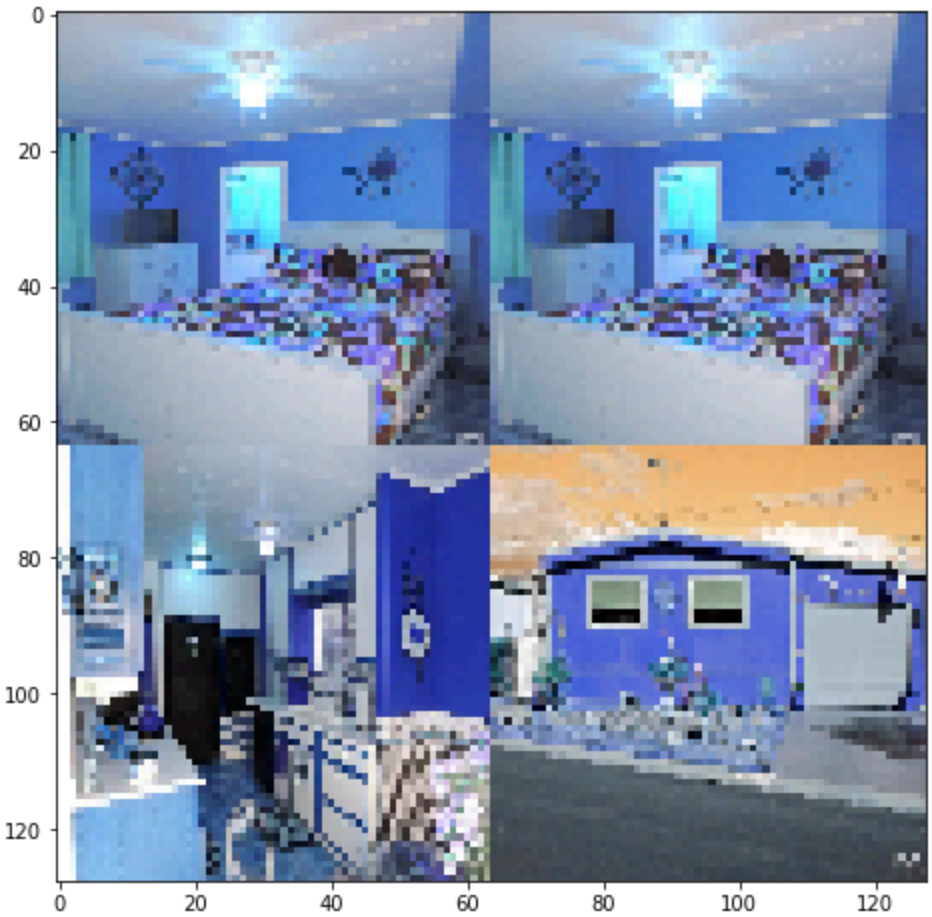

0
20
40
60
80
100
120
0
20
40
60
80
100
120


0
20
40
60
80
100
120
0
20
40
60
80
100
120

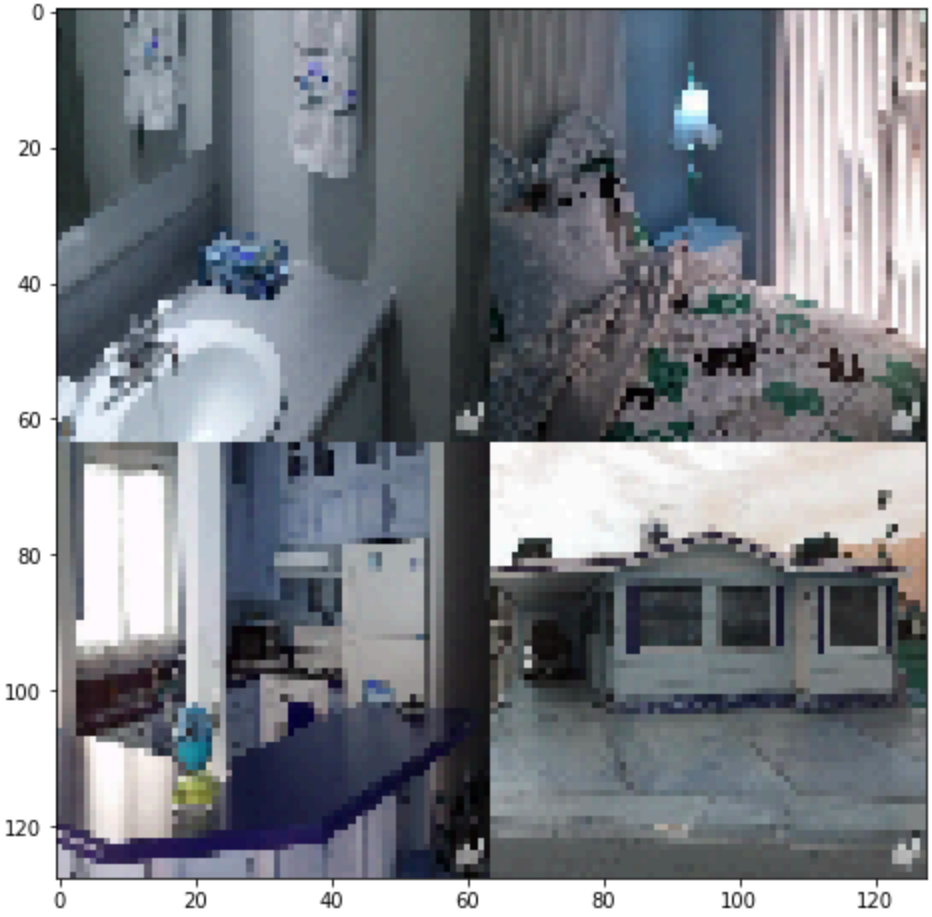

0
20
40
60
80
100
120
0
20
40
60
80
100
120

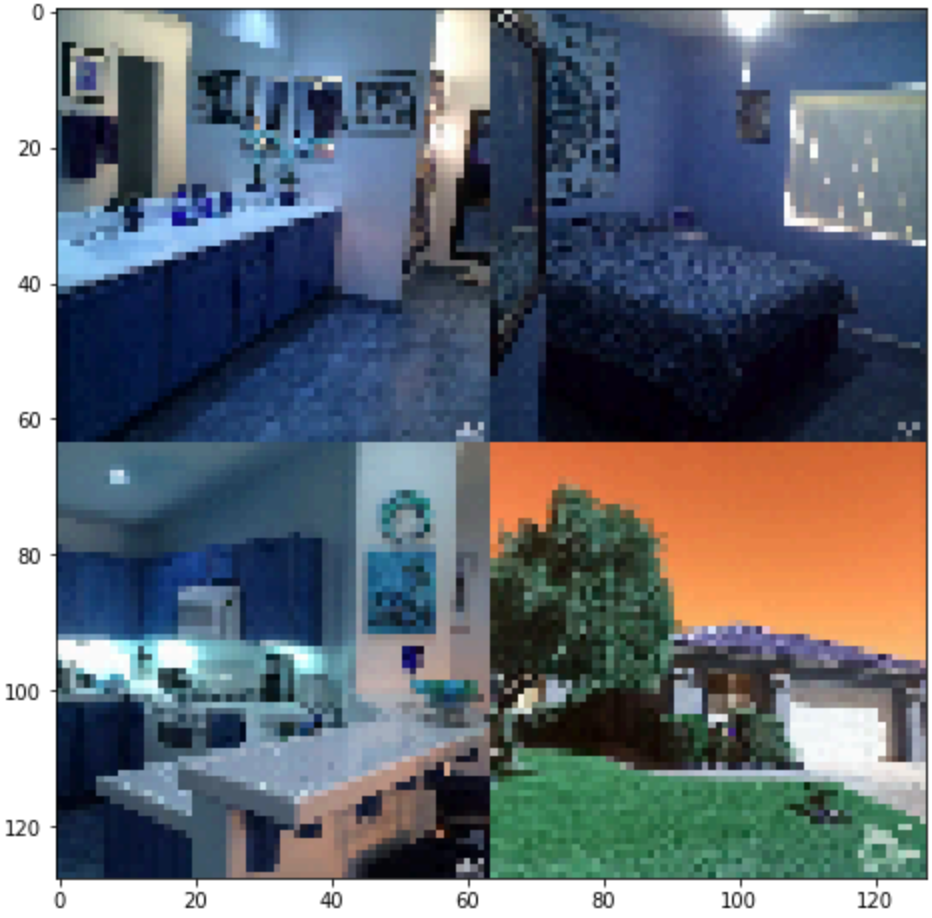

0
20
40
60
80
100
120
0
20
40
60
80
100
120

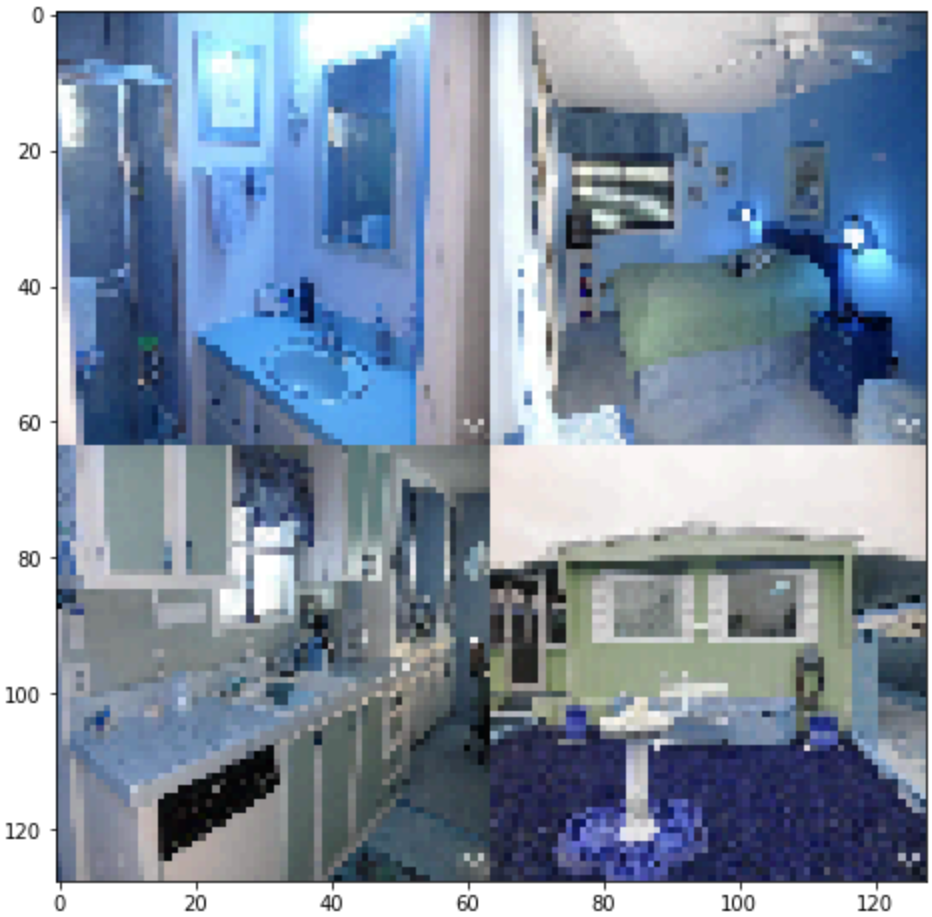

0
20
40
60
80
100
120
0
20
40
60
80
100
120


0
20
40
60
80
100
120
0
20
40
60
80
100
120

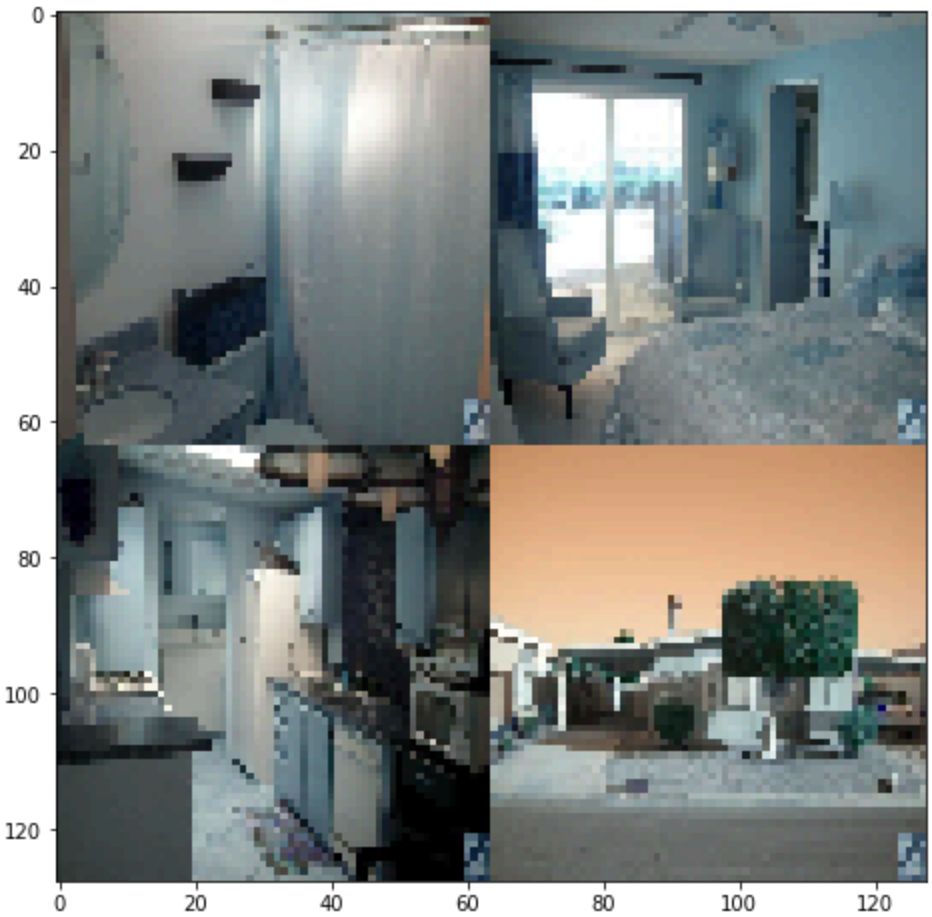

0
20
40
60
80
100
120
0
20
40
60
80
100
120

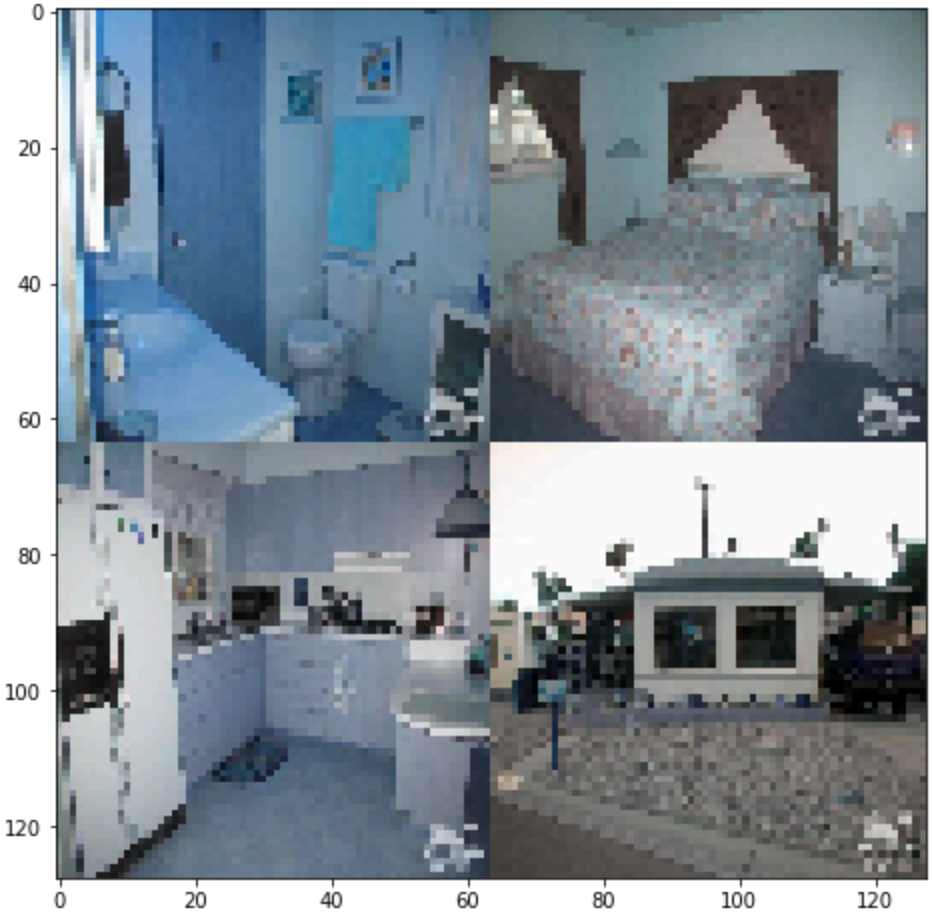

0
20
40
60
80
100
120
0
20
40
60
80
100
120


0
20
40
60
80
100
120
0
20
40
60
80
100
120

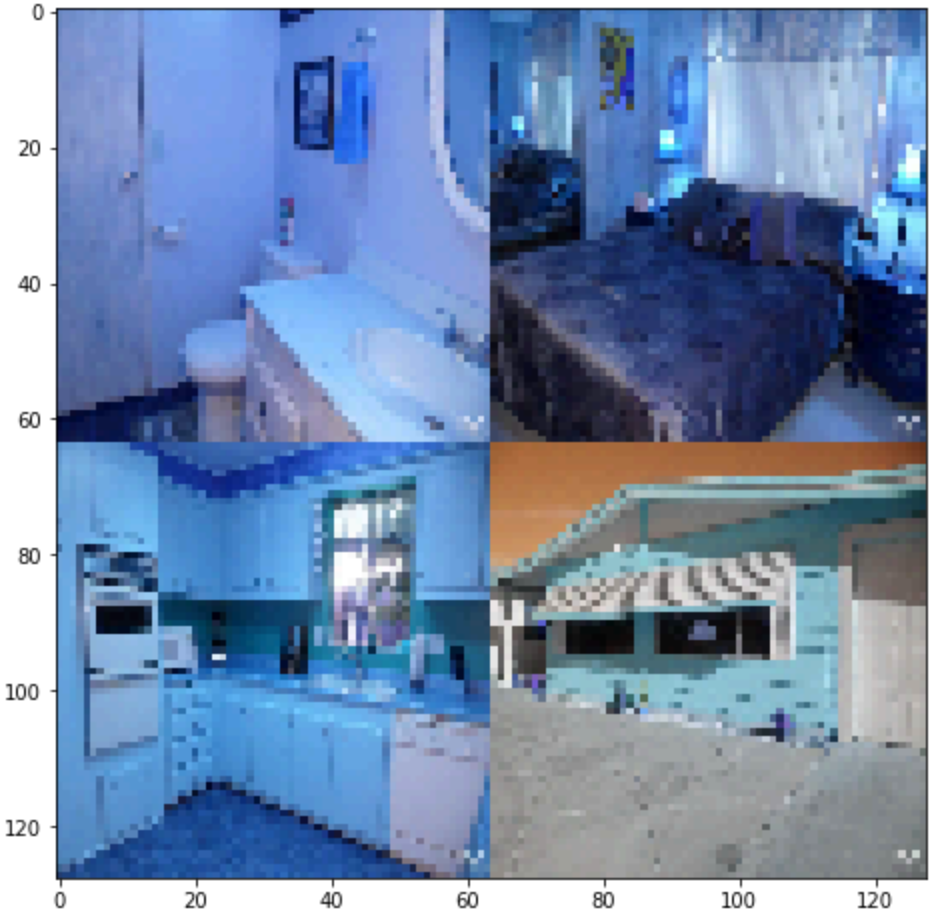

0
20
40
60
80
100
120
0
20
40
60
80
100
120

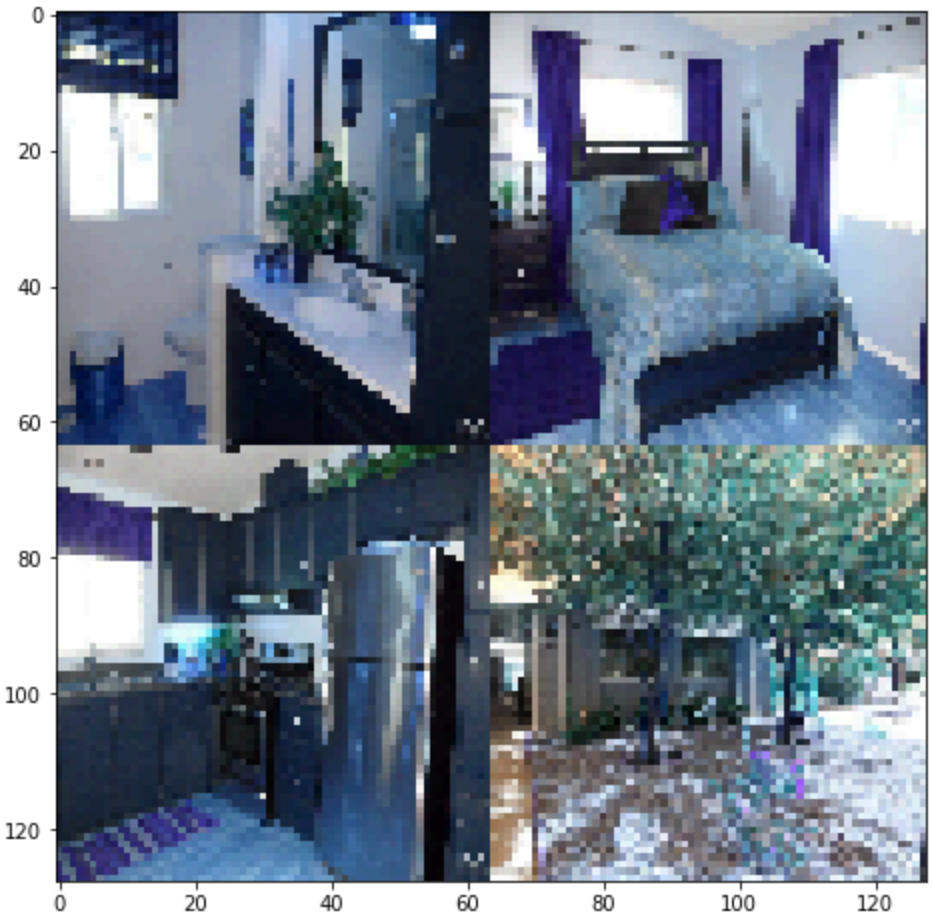

0
20
40
60
80
100
120
0
20
40
60
80
100
120

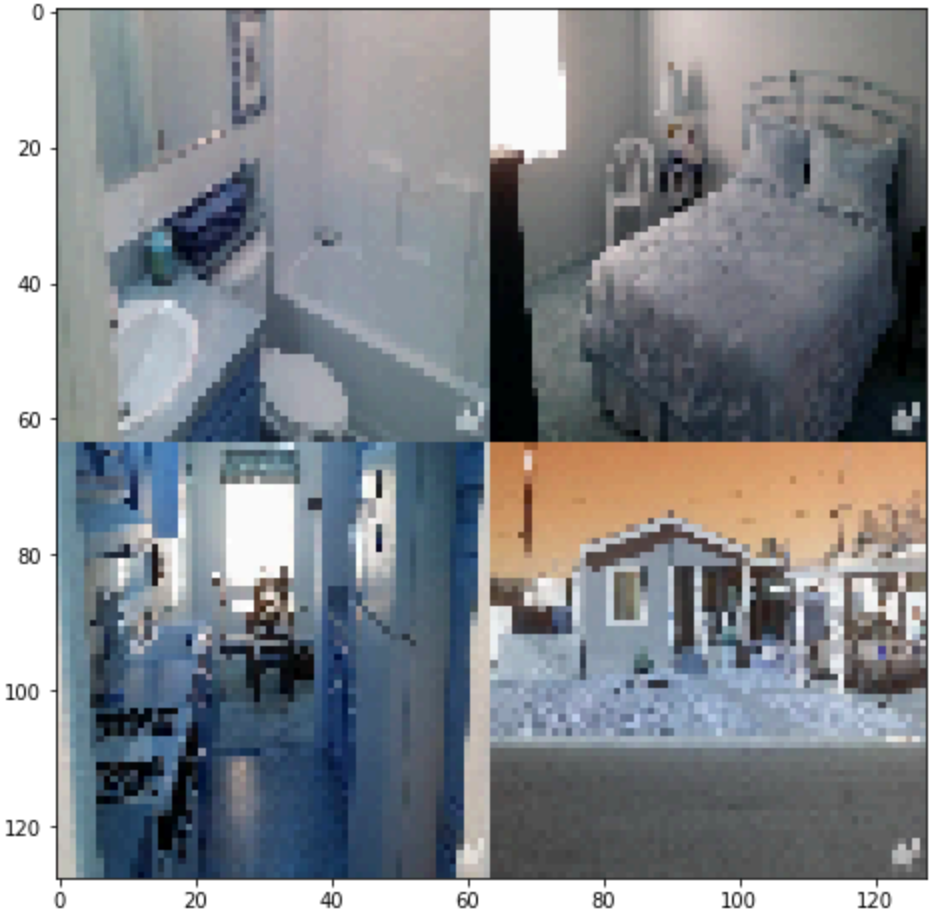

0
20
40
60
80
100
120
0
20
40
60
80
100
120


0
20
40
60
80
100
120
0
20
40
60
80
100
120

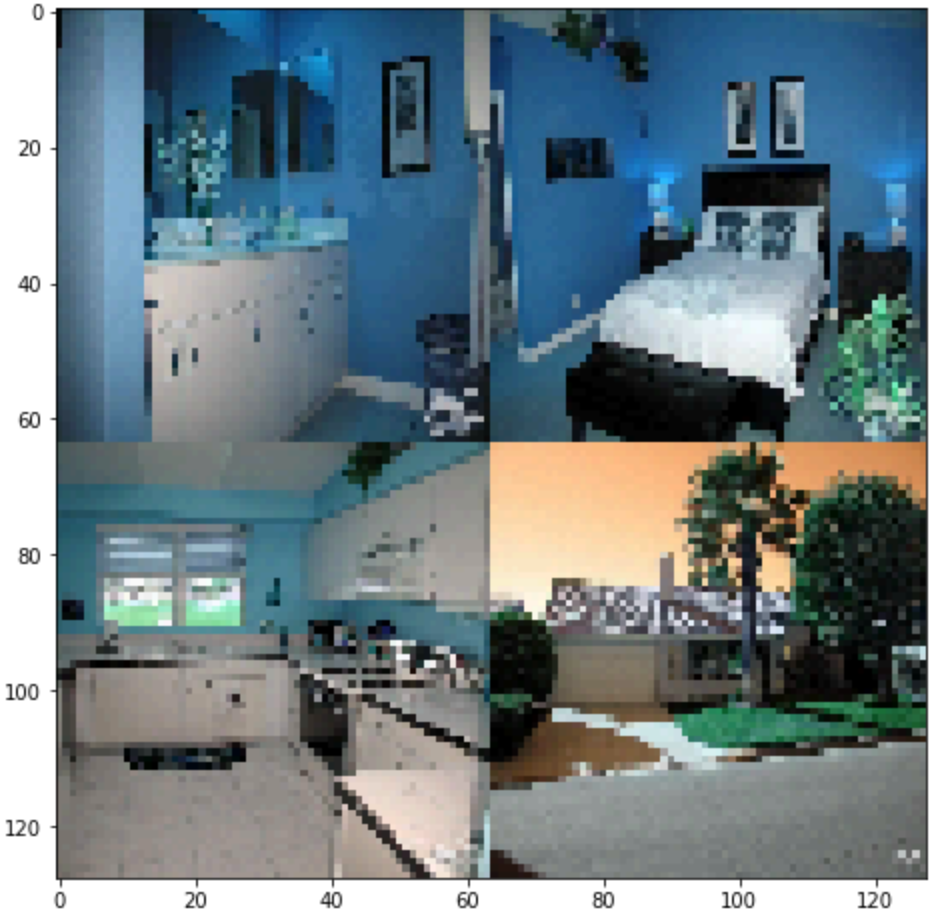

0
20
40
60
80
100
120
0
20
40
60
80
100
120


0
20
40
60
80
100
120
0
20
40
60
80
100
120


0
20
40
60
80
100
120
0
20
40
60
80
100
120

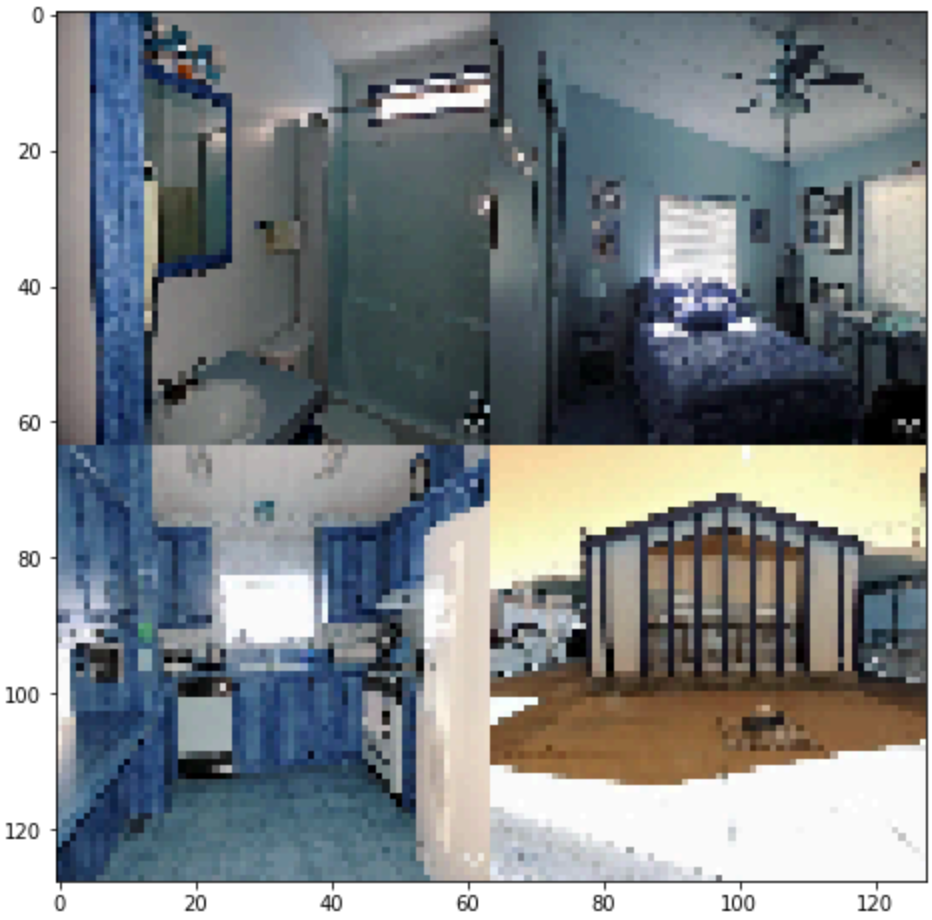

0
20
40
60
80
100
120
0
20
40
60
80
100
120

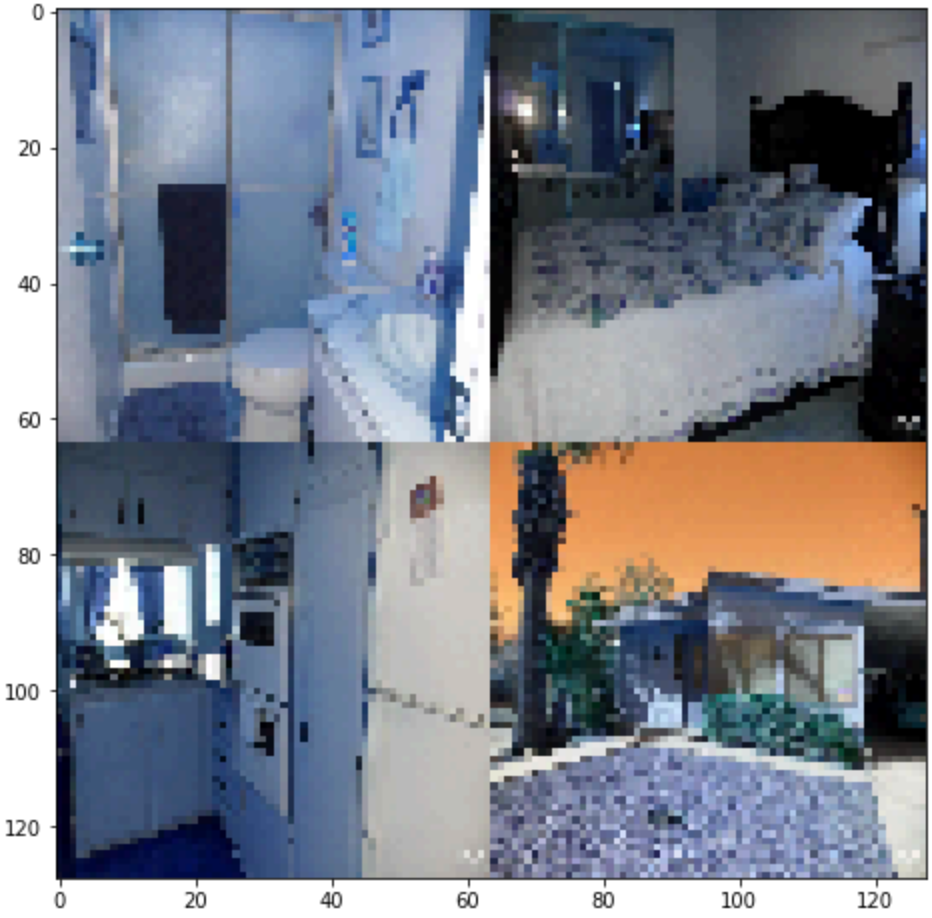

0
20
40
60
80
100
120
0
20
40
60
80
100
120


0
20
40
60
80
100
120
0
20
40
60
80
100
120


0
20
40
60
80
100
120
0
20
40
60
80
100
120


0
20
40
60
80
100
120
0
20
40
60
80
100
120

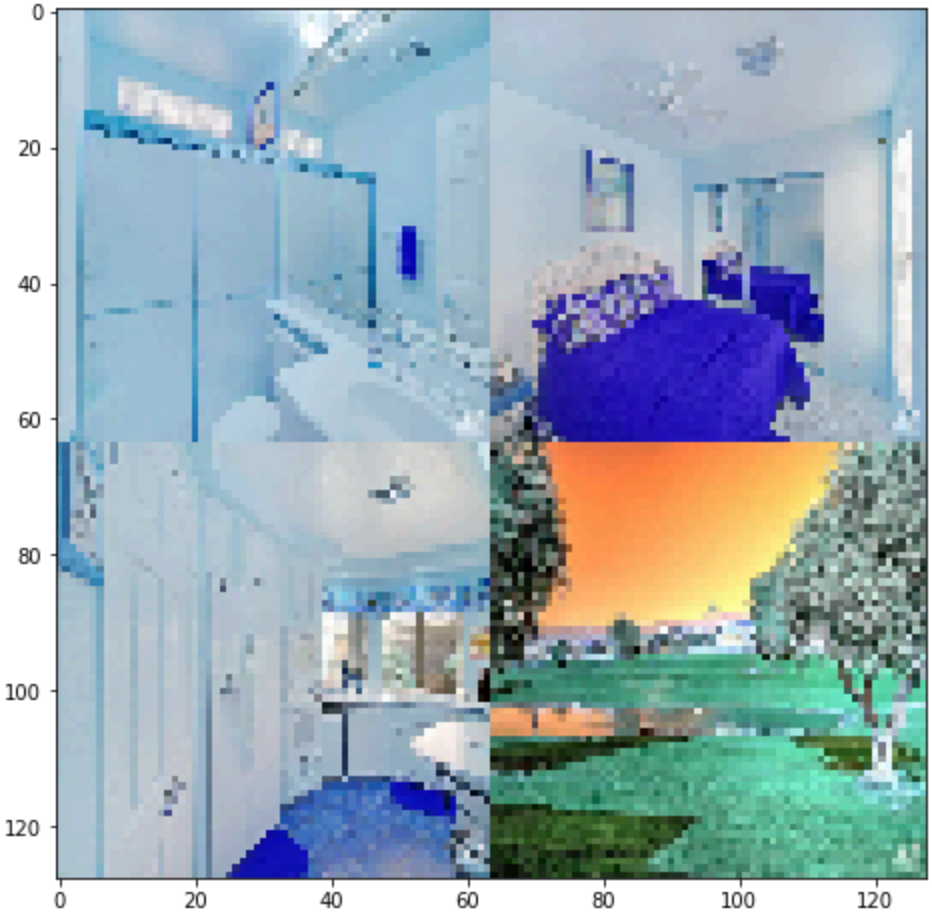

0
20
40
60
80
100
120
0
20
40
60
80
100
120


0
20
40
60
80
100
120
0
20
40
60
80
100
120


0
20
40
60
80
100
120
0
20
40
60
80
100
120


0
20
40
60
80
100
120
0
20
40
60
80
100
120


0
20
40
60
80
100
120
0
20
40
60
80
100
120


0
20
40
60
80
100
120
0
20
40
60
80
100
120


0
20
40
60
80
100
120
0
20
40
60
80
100
120


0
20
40
60
80
100
120
0
20
40
60
80
100
120


0
20
40
60
80
100
120
0
20
40
60
80
100
120

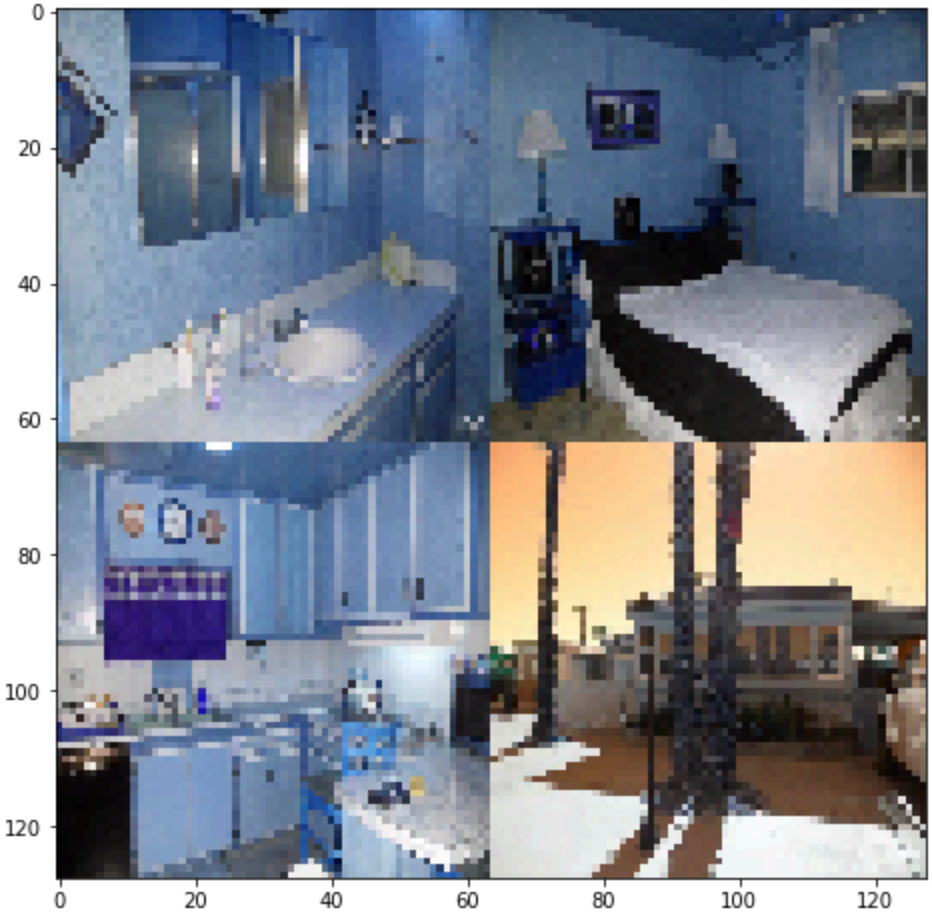

0
20
40
60
80
100
120
0
20
40
60
80
100
120


0
20
40
60
80
100
120
0
20
40
60
80
100
120


0
20
40
60
80
100
120
0
20
40
60
80
100
120


0
20
40
60
80
100
120
0
20
40
60
80
100
120

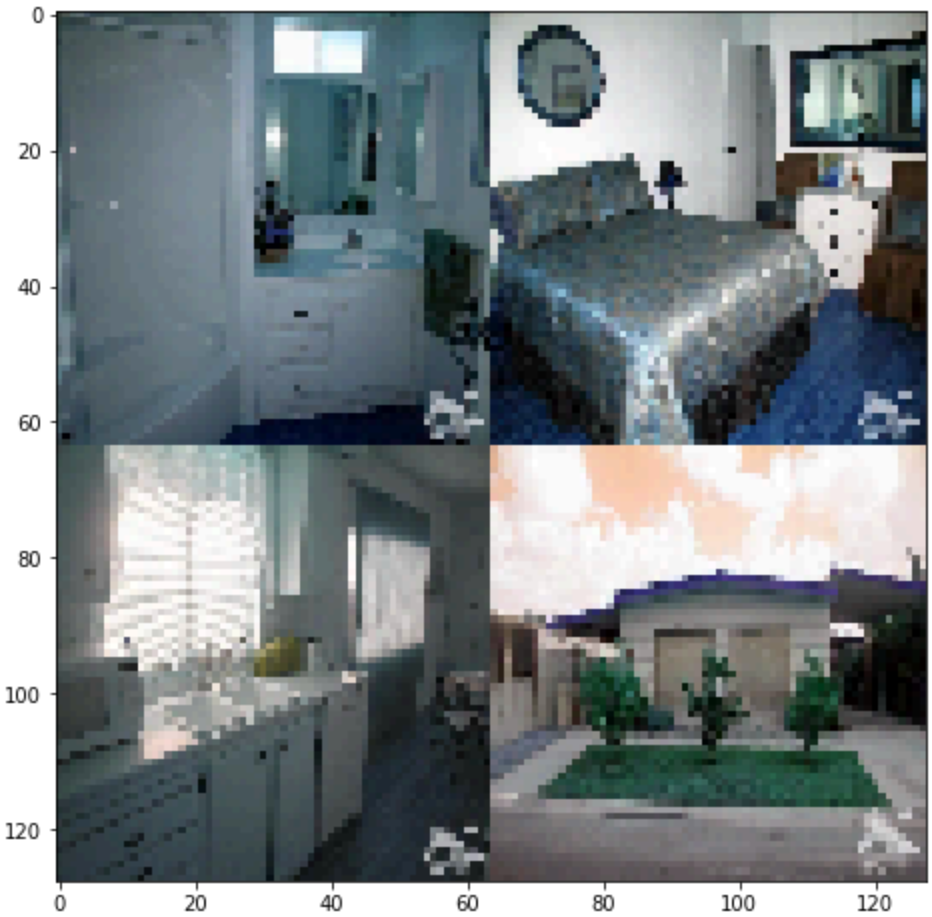

0
20
40
60
80
100
120
0
20
40
60
80
100
120


0
20
40
60
80
100
120
0
20
40
60
80
100
120


0
20
40
60
80
100
120
0
20
40
60
80
100
120

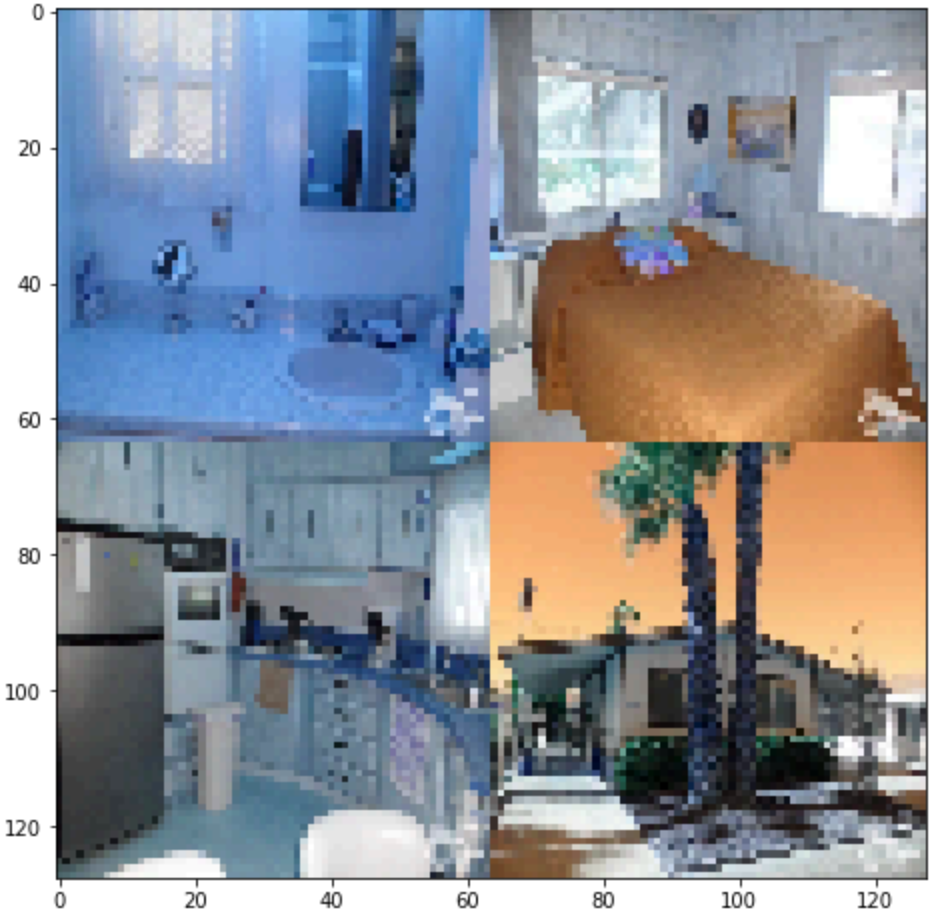

0
20
40
60
80
100
120
0
20
40
60
80
100
120


0
20
40
60
80
100
120
0
20
40
60
80
100
120

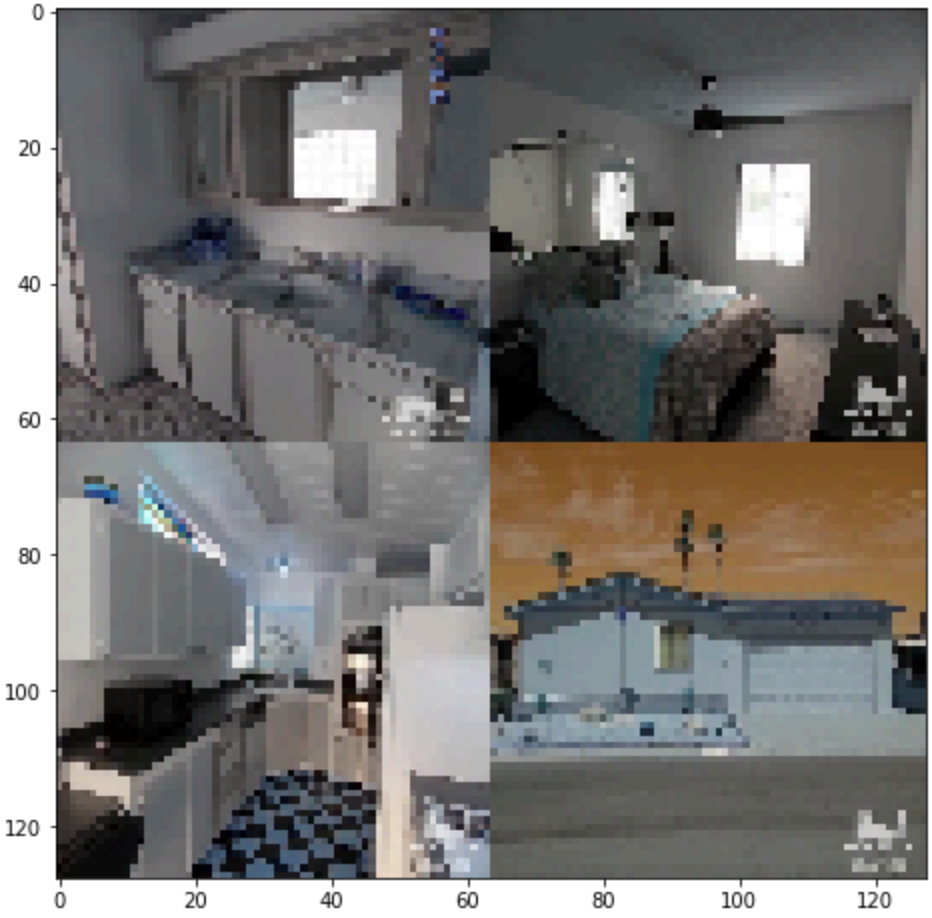

0
20
40
60
80
100
120
0
20
40
60
80
100
120

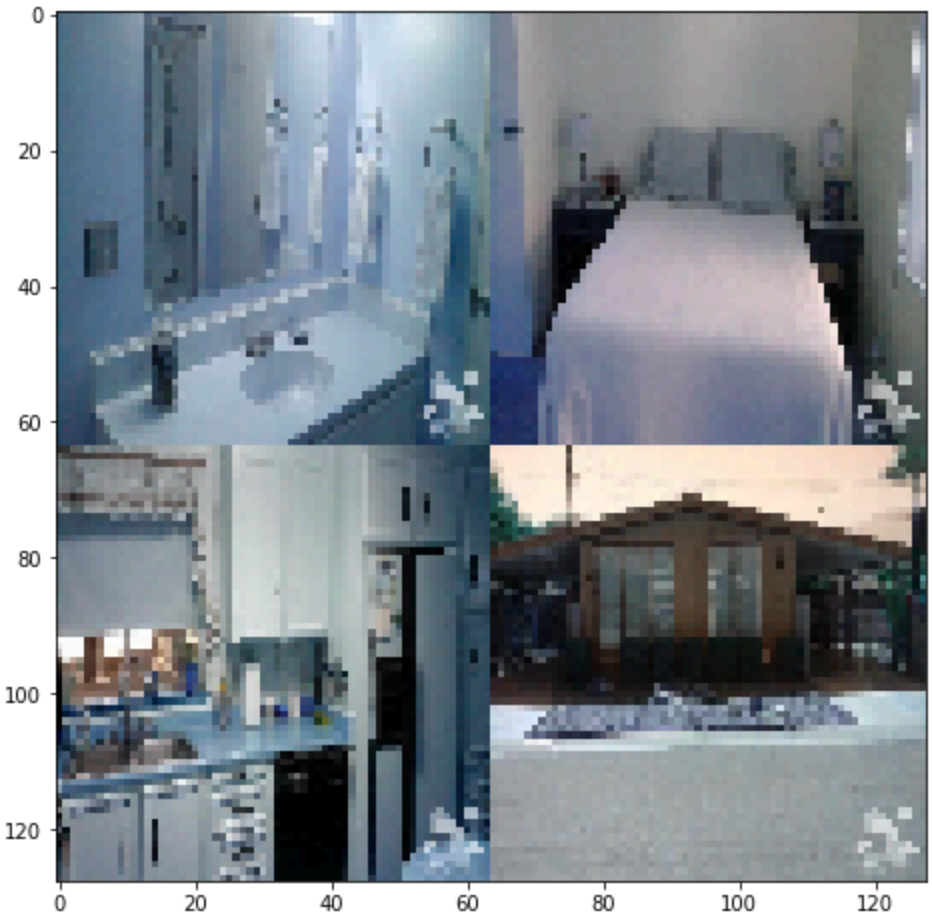

0
20
40
60
80
100
120
0
20
40
60
80
100
120


0
20
40
60
80
100
120
0
20
40
60
80
100
120


0
20
40
60
80
100
120
0
20
40
60
80
100
120


0
20
40
60
80
100
120
0
20
40
60
80
100
120

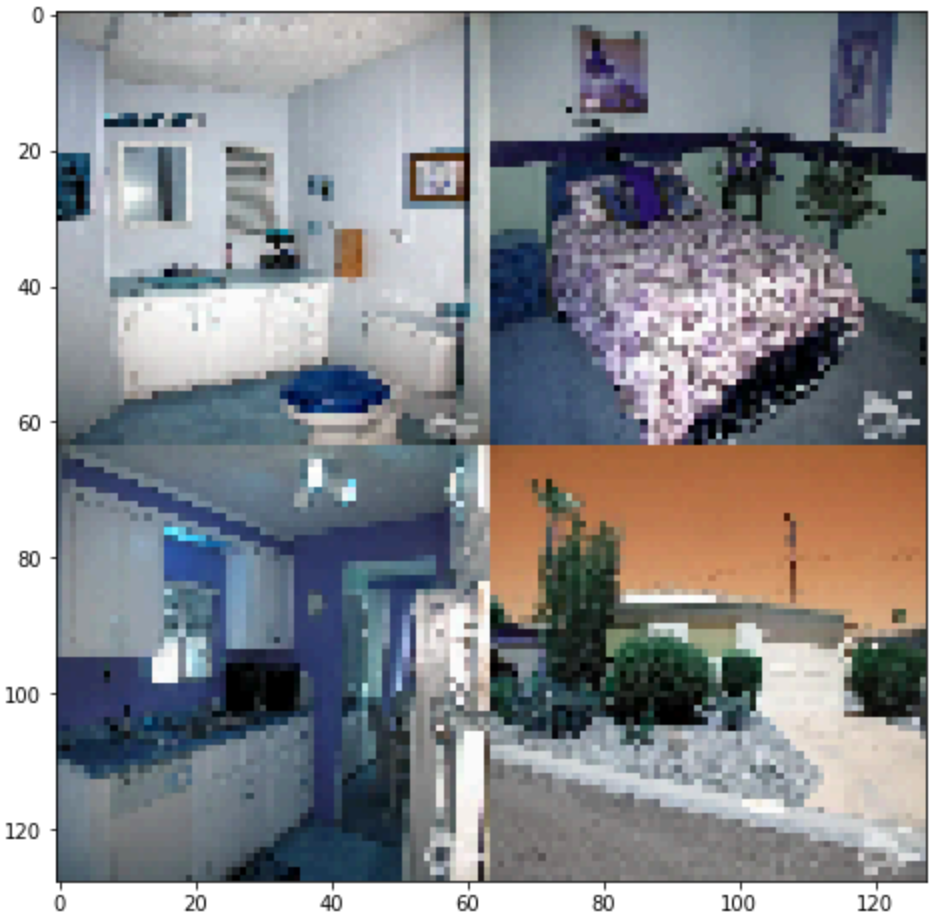

0
20
40
60
80
100
120
0
20
40
60
80
100
120


0
20
40
60
80
100
120
0
20
40
60
80
100
120


0
20
40
60
80
100
120
0
20
40
60
80
100
120


0
20
40
60
80
100
120
0
20
40
60
80
100
120

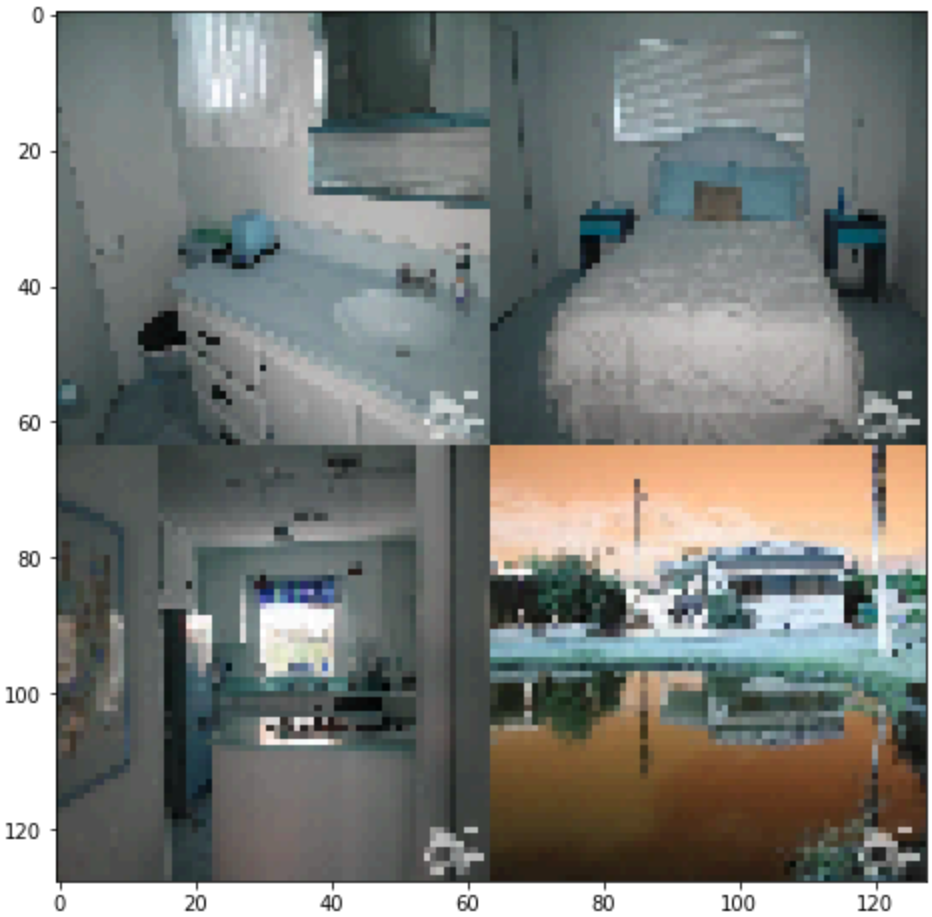

0
20
40
60
80
100
120
0
20
40
60
80
100
120


0
20
40
60
80
100
120
0
20
40
60
80
100
120


0
20
40
60
80
100
120
0
20
40
60
80
100
120


0
20
40
60
80
100
120
0
20
40
60
80
100
120


0
20
40
60
80
100
120
0
20
40
60
80
100
120

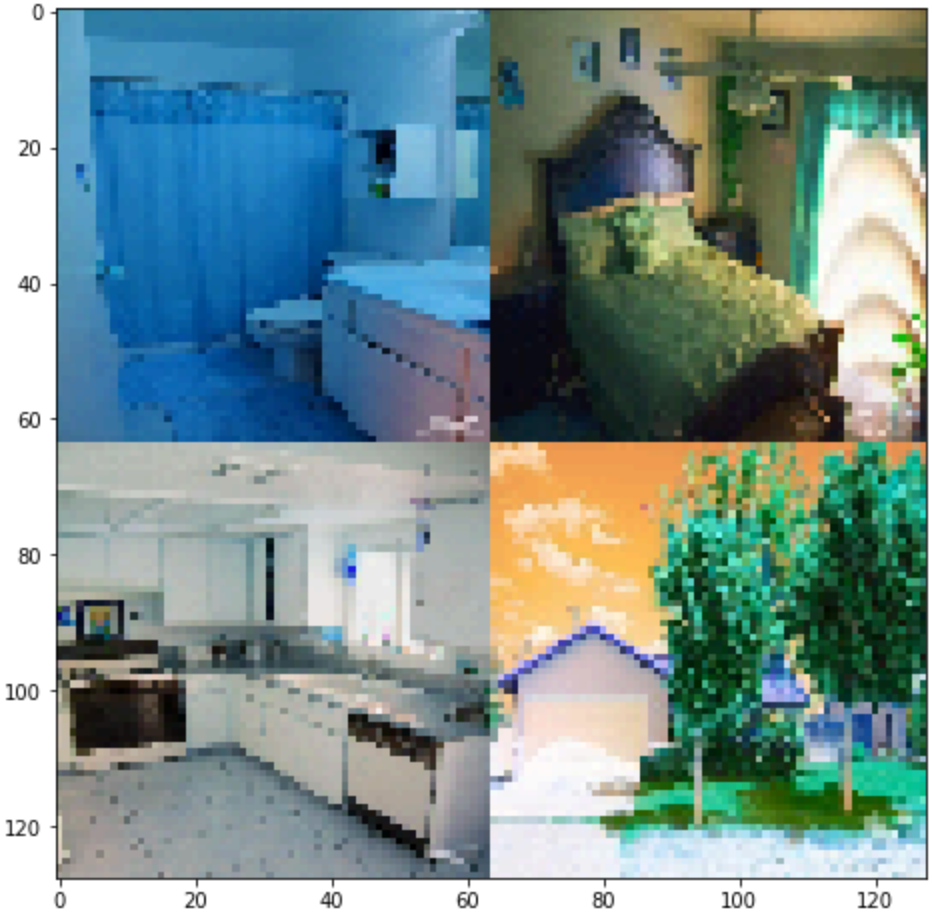

0
20
40
60
80
100
120
0
20
40
60
80
100
120


0
20
40
60
80
100
120
0
20
40
60
80
100
120


0
20
40
60
80
100
120
0
20
40
60
80
100
120


0
20
40
60
80
100
120
0
20
40
60
80
100
120

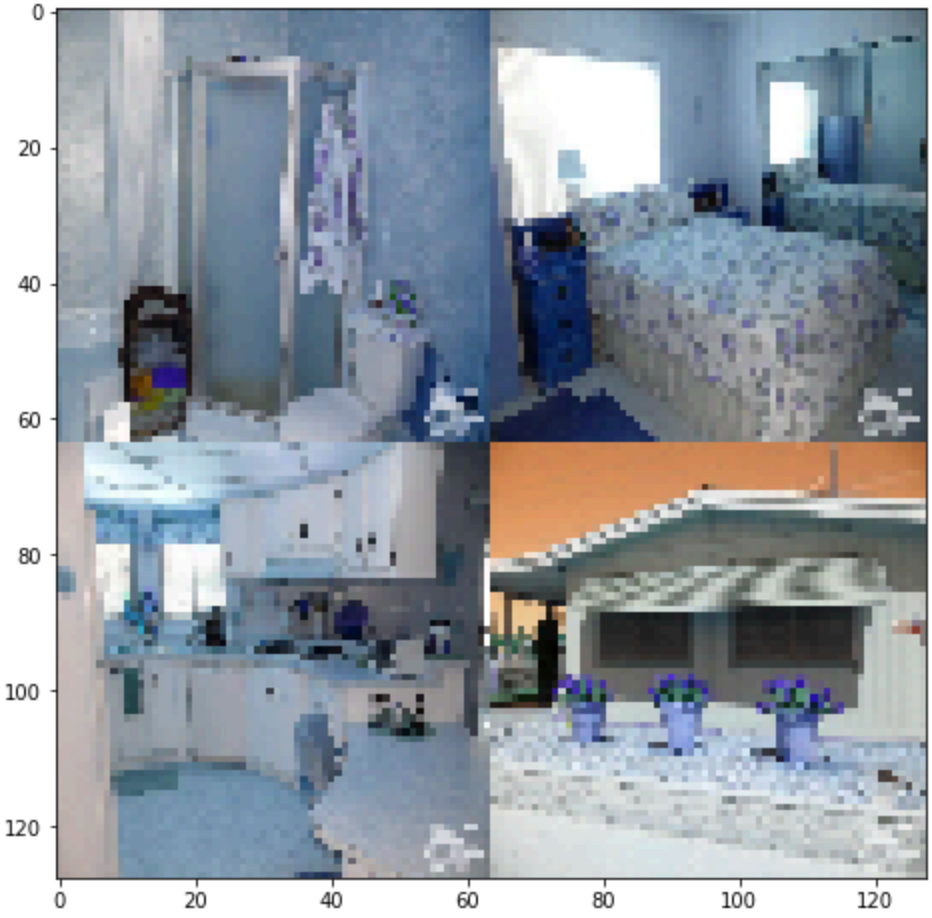

0
20
40
60
80
100
120
0
20
40
60
80
100
120

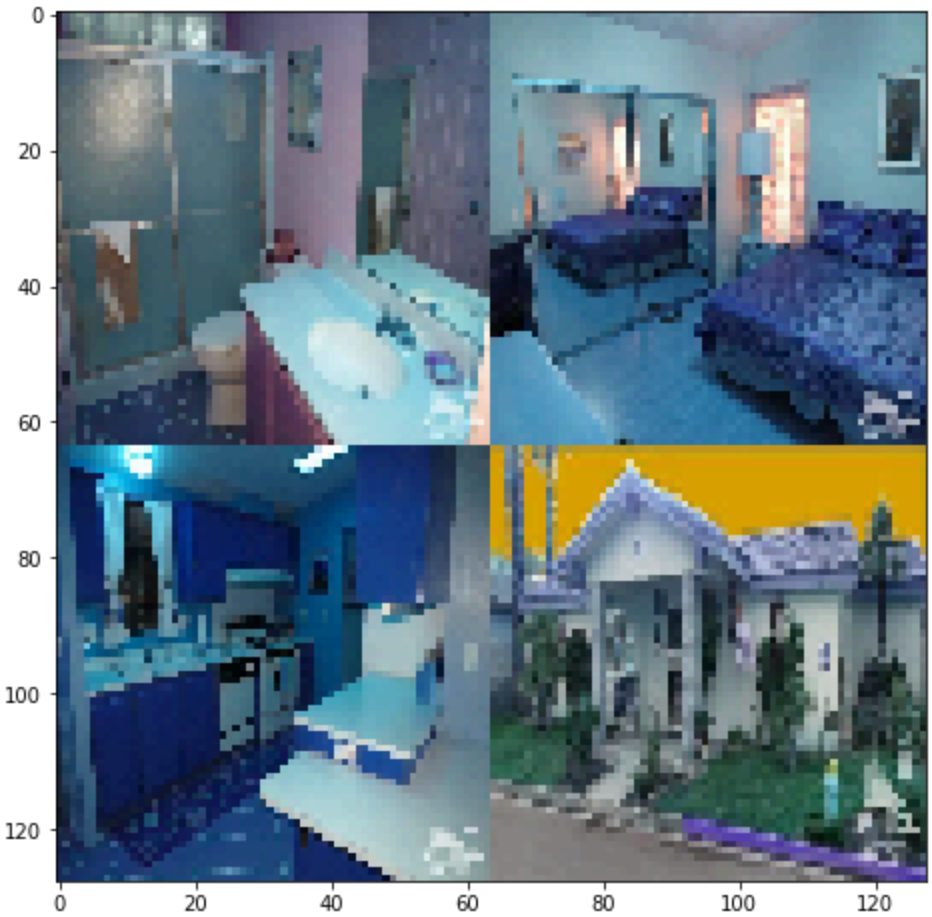

0
20
40
60
80
100
120
0
20
40
60
80
100
120

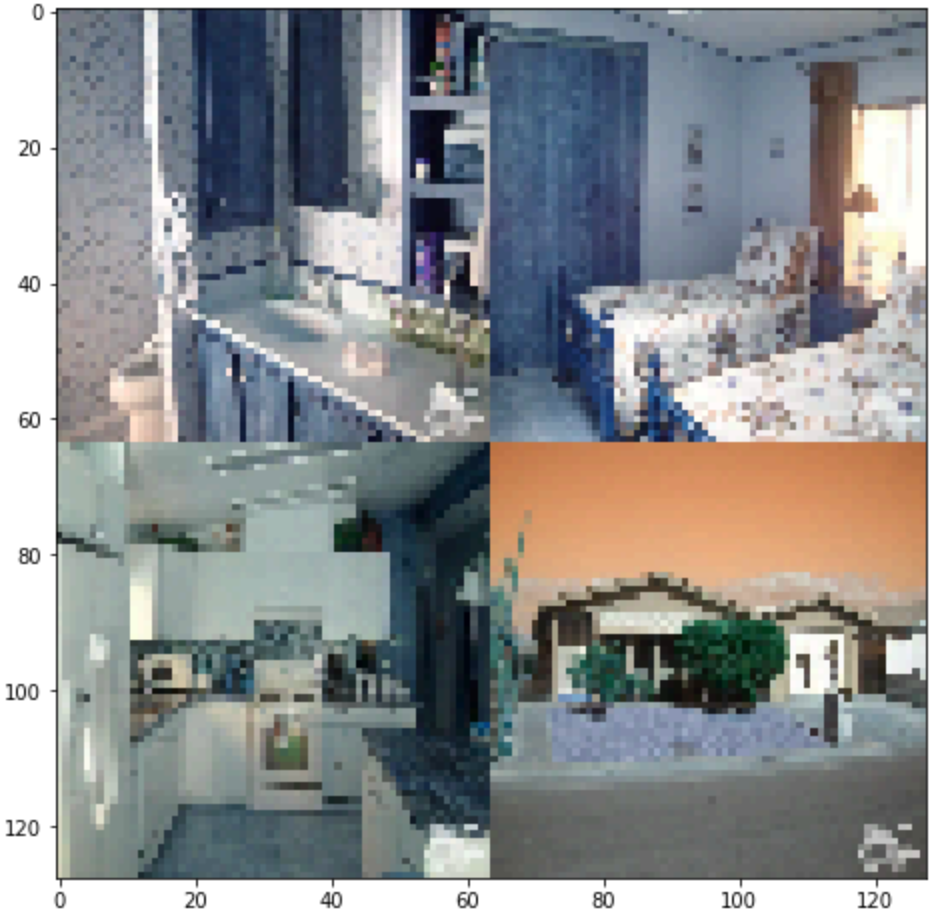

0
20
40
60
80
100
120
0
20
40
60
80
100
120


0
20
40
60
80
100
120
0
20
40
60
80
100
120

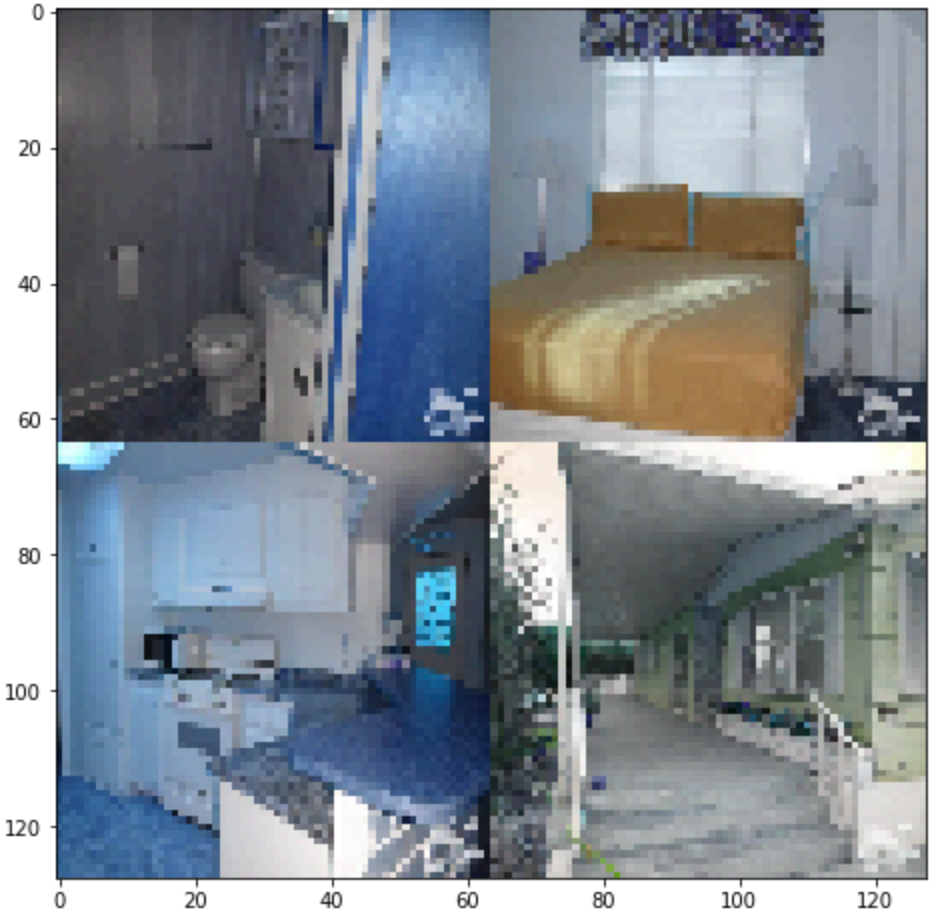

0
20
40
60
80
100
120
0
20
40
60
80
100
120


0
20
40
60
80
100
120
0
20
40
60
80
100
120


0
20
40
60
80
100
120
0
20
40
60
80
100
120


0
20
40
60
80
100
120
0
20
40
60
80
100
120

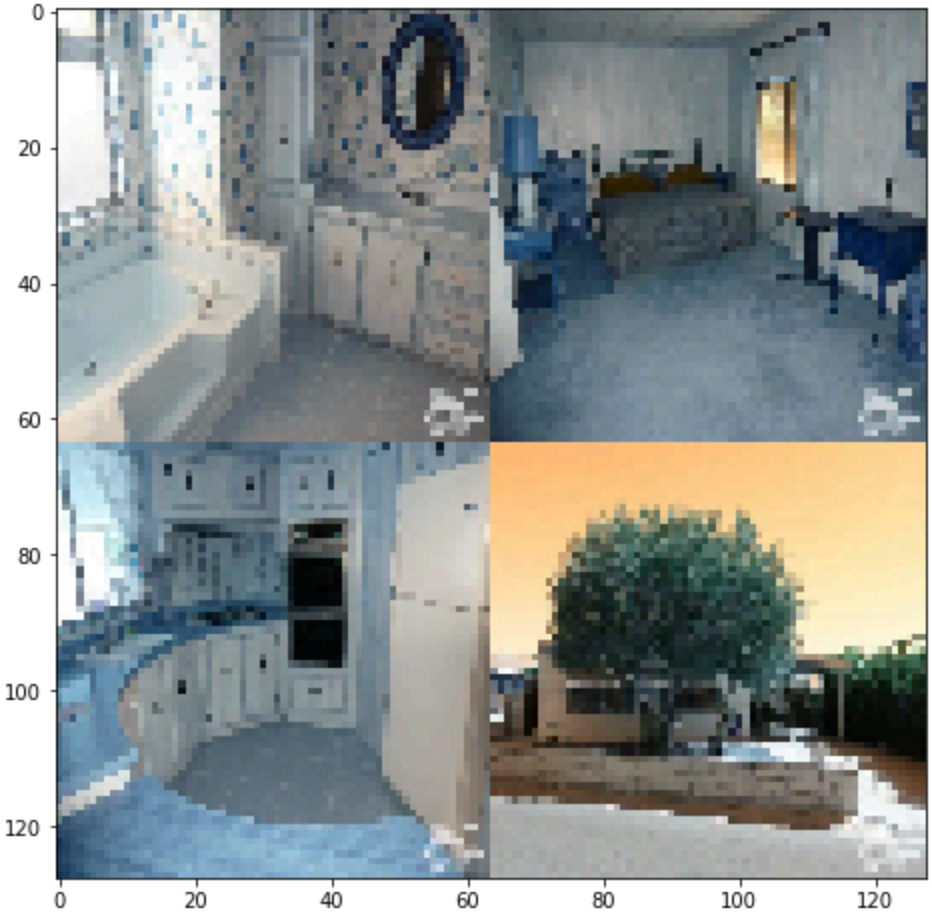

0
20
40
60
80
100
120
0
20
40
60
80
100
120

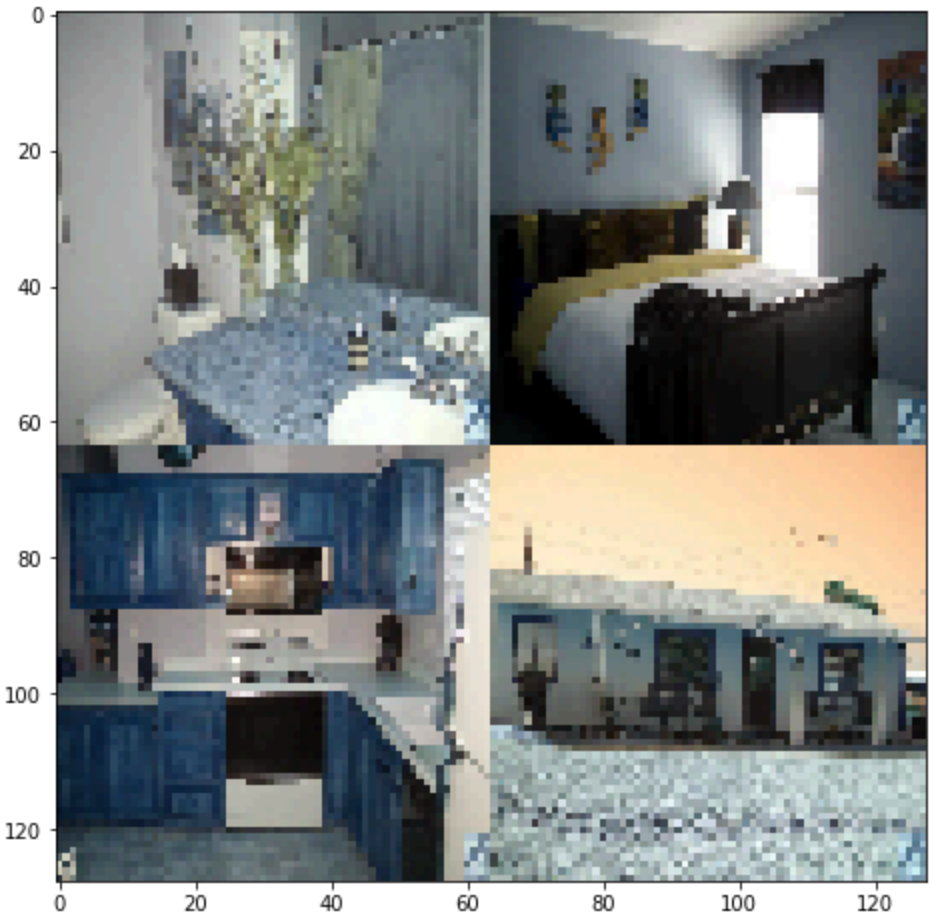

0
20
40
60
80
100
120
0
20
40
60
80
100
120

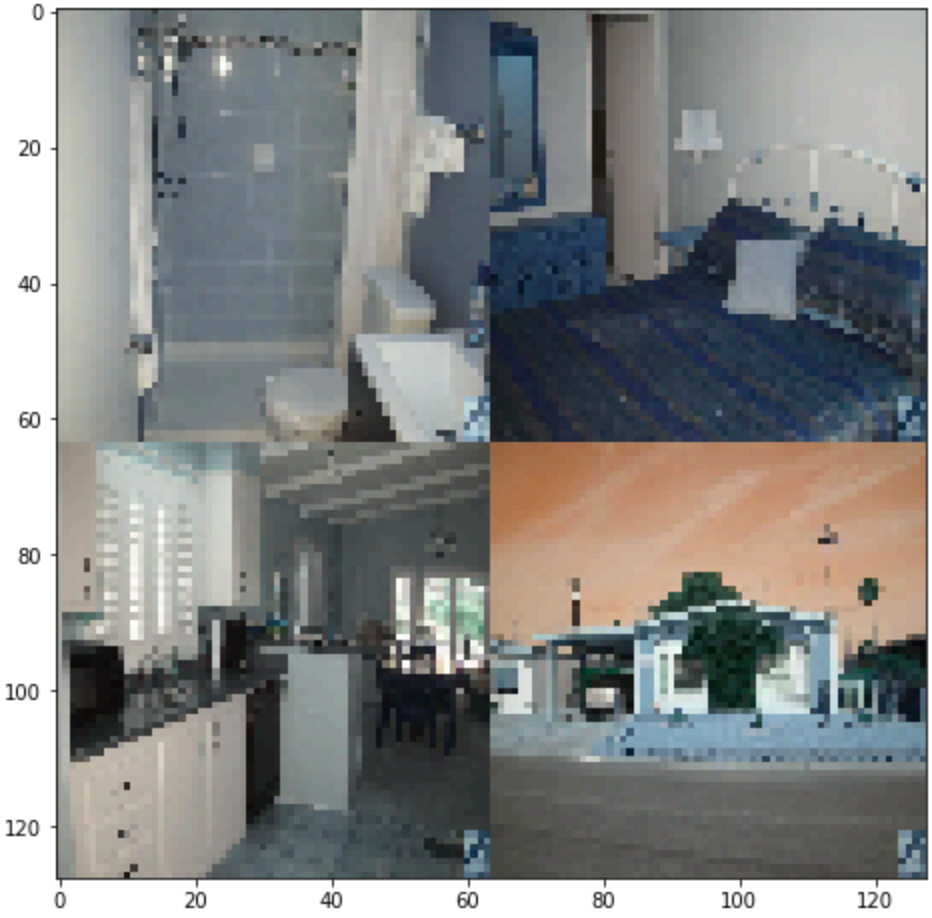

0
20
40
60
80
100
120
0
20
40
60
80
100
120


0
20
40
60
80
100
120
0
20
40
60
80
100
120


0
20
40
60
80
100
120
0
20
40
60
80
100
120


0
20
40
60
80
100
120
0
20
40
60
80
100
120


0
20
40
60
80
100
120
0
20
40
60
80
100
120


0
20
40
60
80
100
120
0
20
40
60
80
100
120

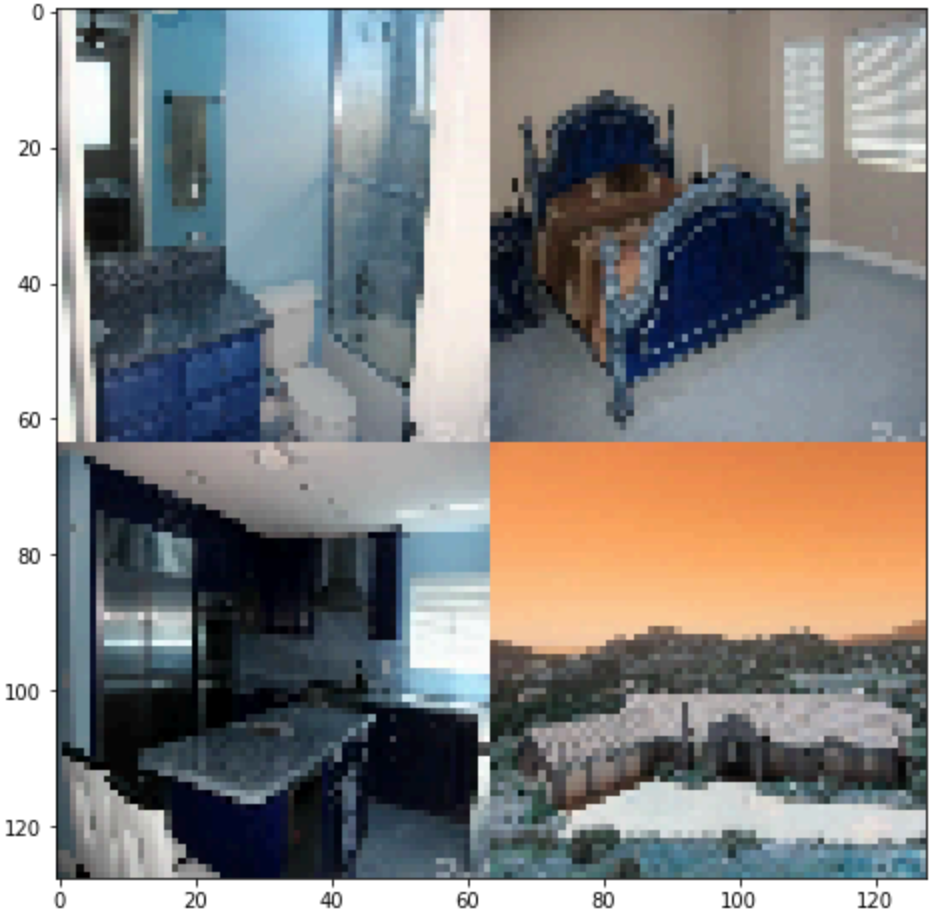

0
20
40
60
80
100
120
0
20
40
60
80
100
120


0
20
40
60
80
100
120
0
20
40
60
80
100
120


0
20
40
60
80
100
120
0
20
40
60
80
100
120


0
20
40
60
80
100
120
0
20
40
60
80
100
120


0
20
40
60
80
100
120
0
20
40
60
80
100
120


0
20
40
60
80
100
120
0
20
40
60
80
100
120


0
20
40
60
80
100
120
0
20
40
60
80
100
120


0
20
40
60
80
100
120
0
20
40
60
80
100
120

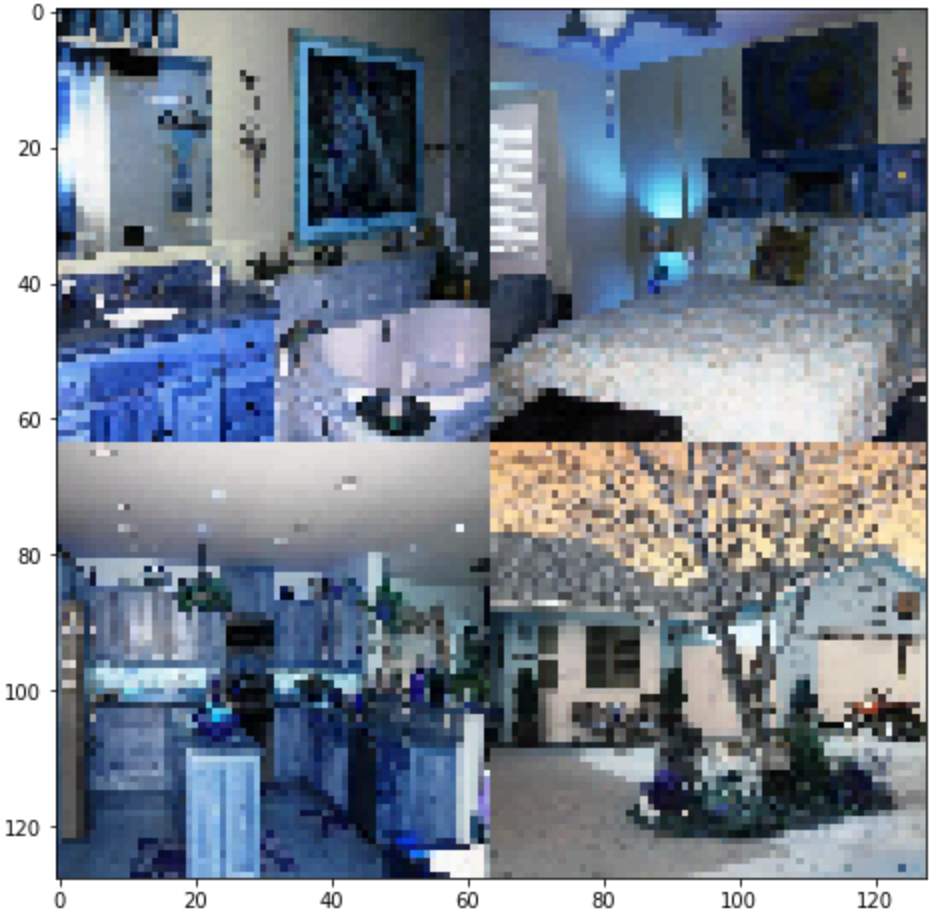

0
20
40
60
80
100
120
0
20
40
60
80
100
120


0
20
40
60
80
100
120
0
20
40
60
80
100
120


0
20
40
60
80
100
120
0
20
40
60
80
100
120

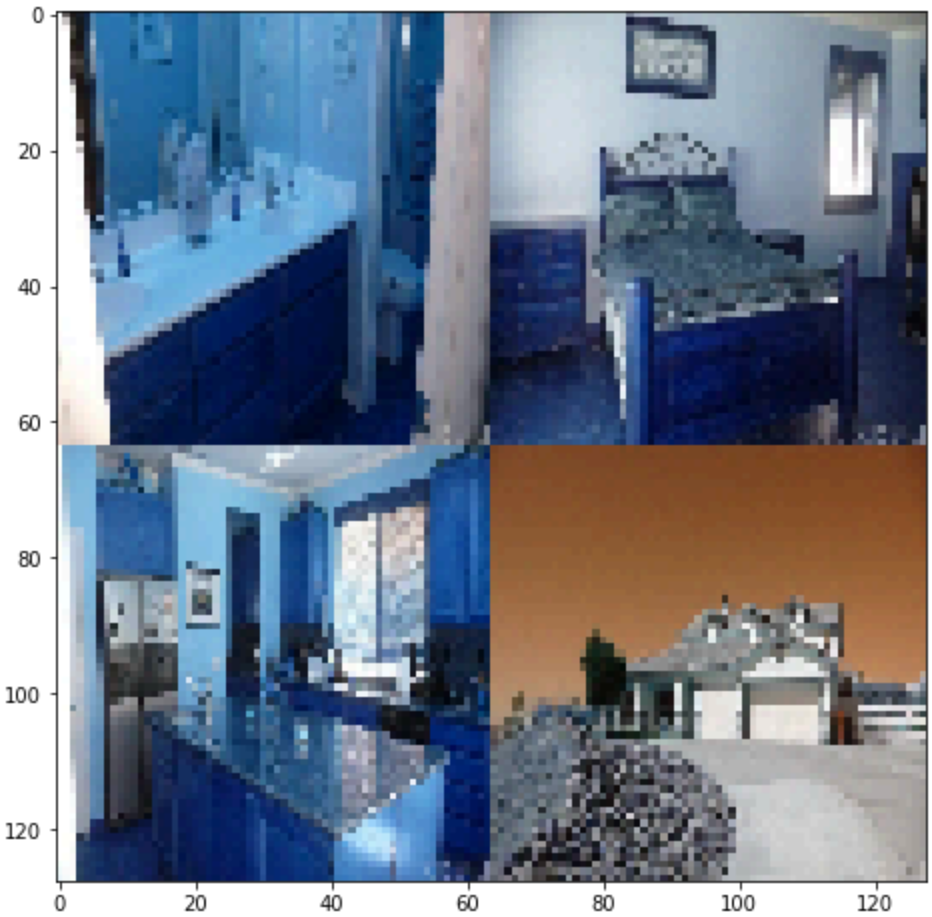

0
20
40
60
80
100
120
0
20
40
60
80
100
120

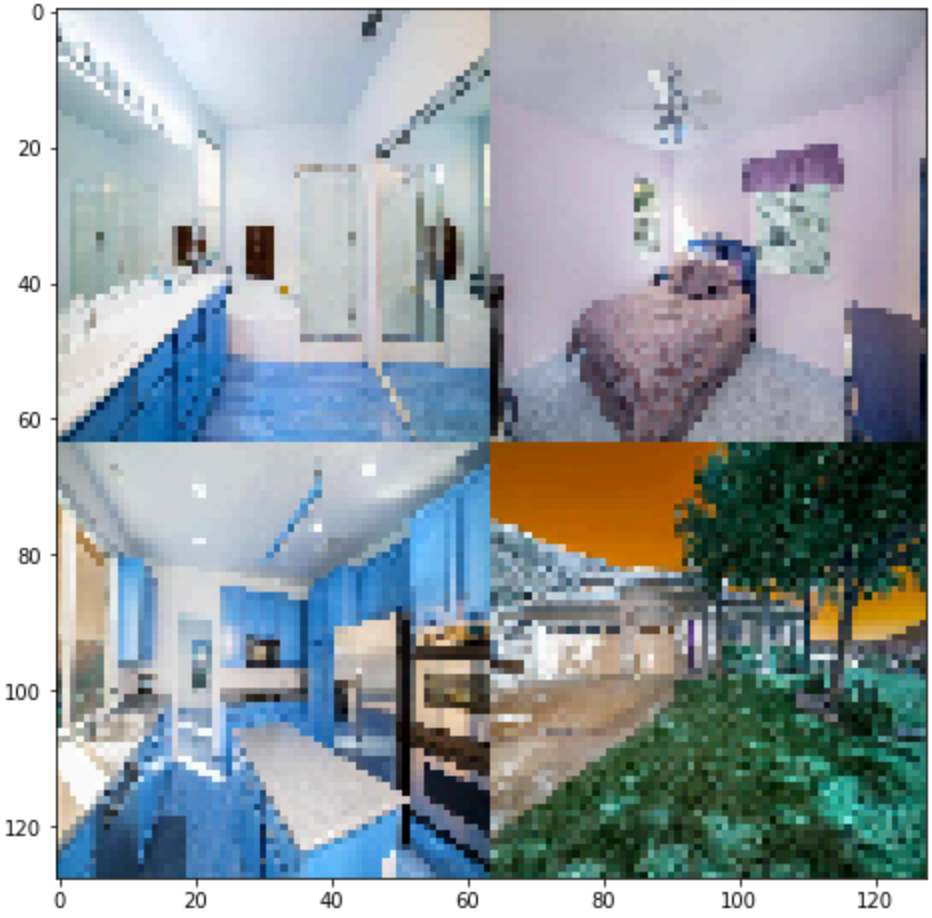

0
20
40
60
80
100
120
0
20
40
60
80
100
120


0
20
40
60
80
100
120
0
20
40
60
80
100
120


0
20
40
60
80
100
120
0
20
40
60
80
100
120


0
20
40
60
80
100
120
0
20
40
60
80
100
120

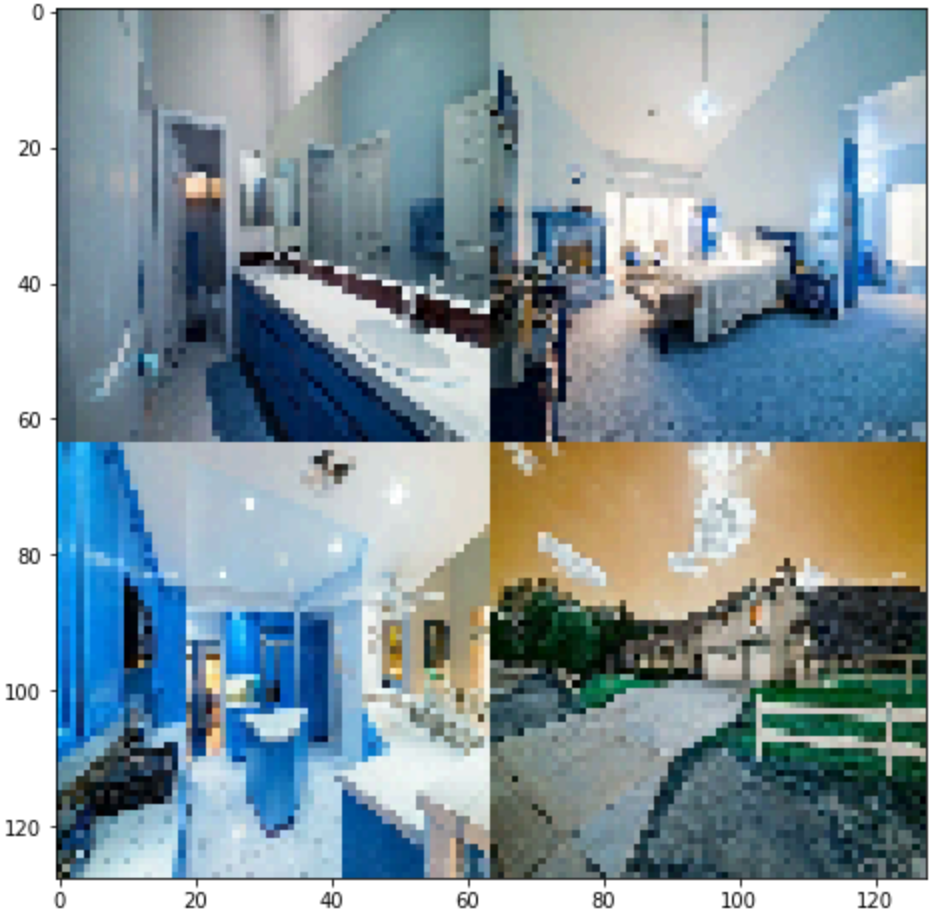

0
20
40
60
80
100
120
0
20
40
60
80
100
120

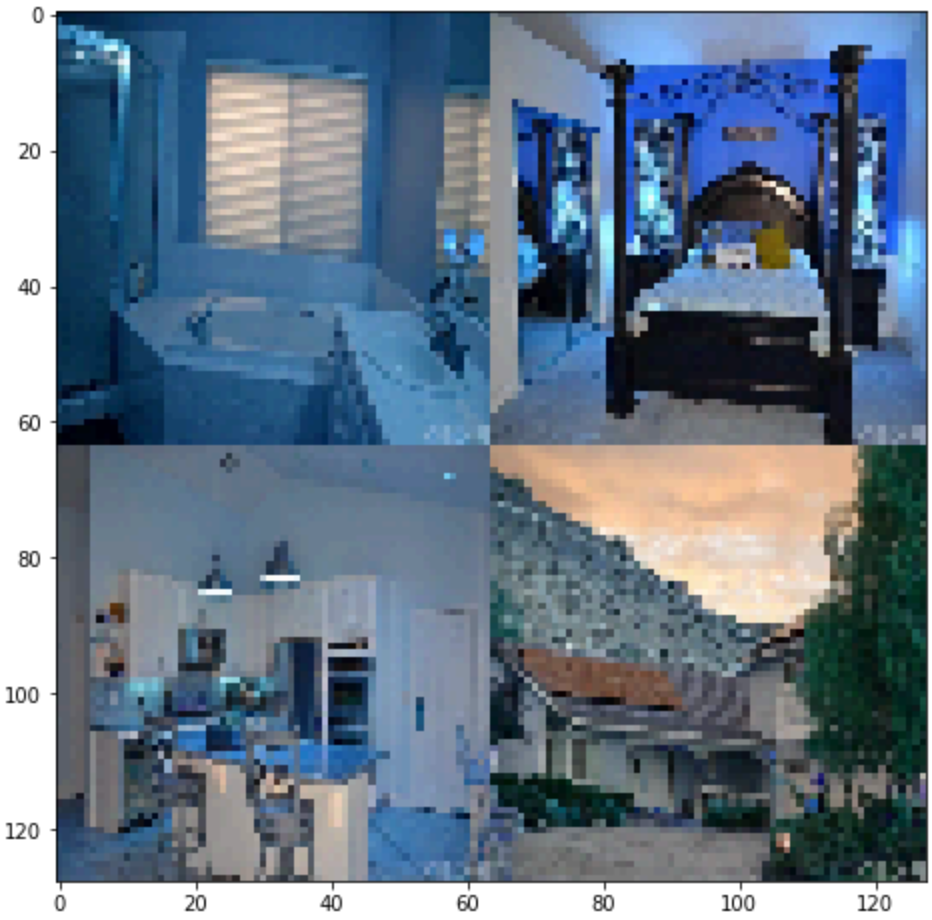

0
20
40
60
80
100
120
0
20
40
60
80
100
120

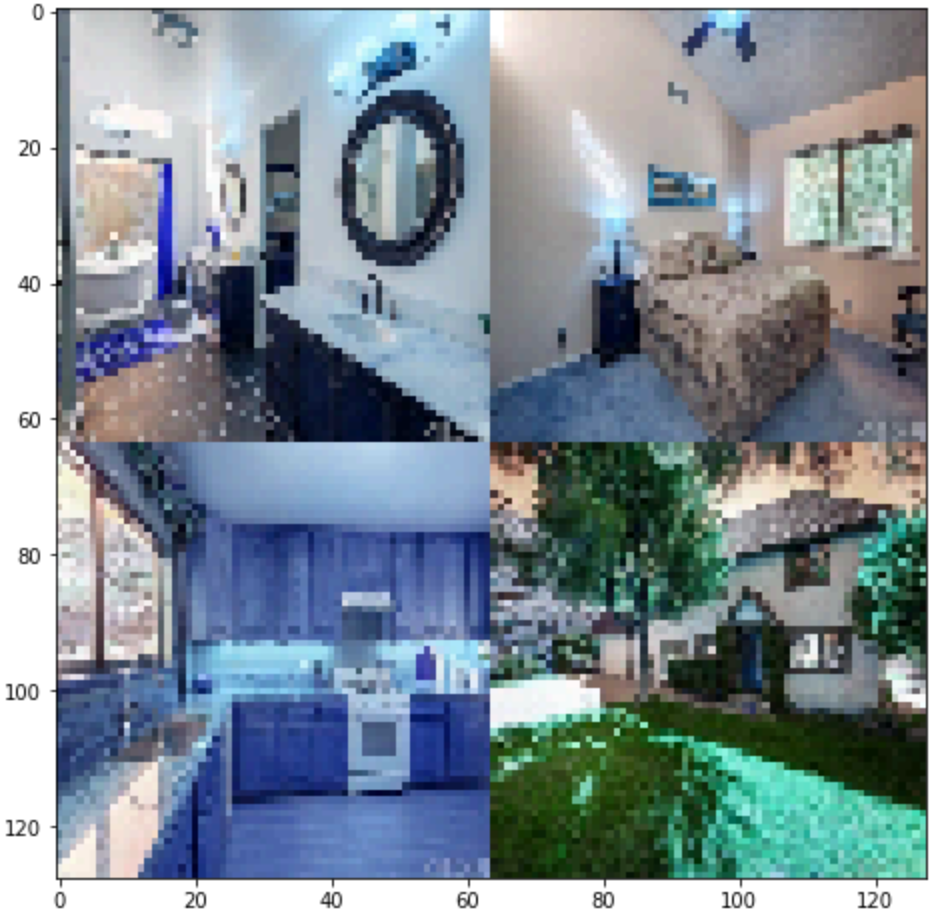

0
20
40
60
80
100
120
0
20
40
60
80
100
120


0
20
40
60
80
100
120
0
20
40
60
80
100
120


0
20
40
60
80
100
120
0
20
40
60
80
100
120


0
20
40
60
80
100
120
0
20
40
60
80
100
120

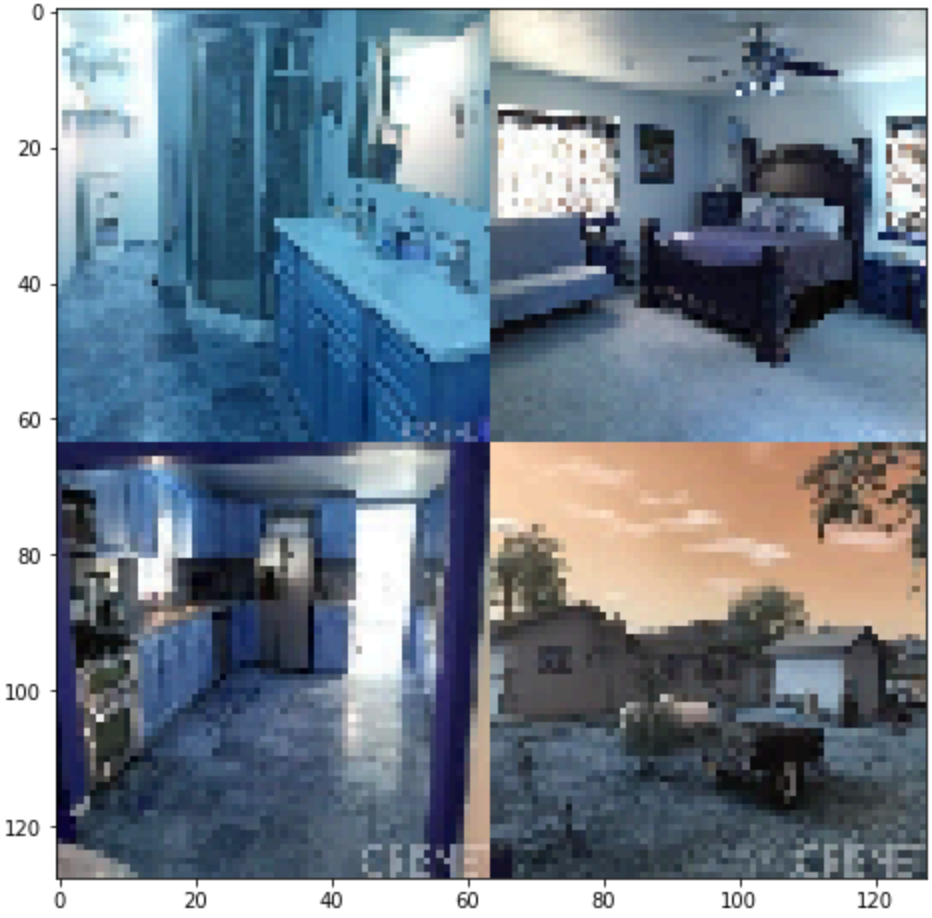

0
20
40
60
80
100
120
0
20
40
60
80
100
120


0
20
40
60
80
100
120
0
20
40
60
80
100
120


0
20
40
60
80
100
120
0
20
40
60
80
100
120


0
20
40
60
80
100
120
0
20
40
60
80
100
120


0
20
40
60
80
100
120
0
20
40
60
80
100
120

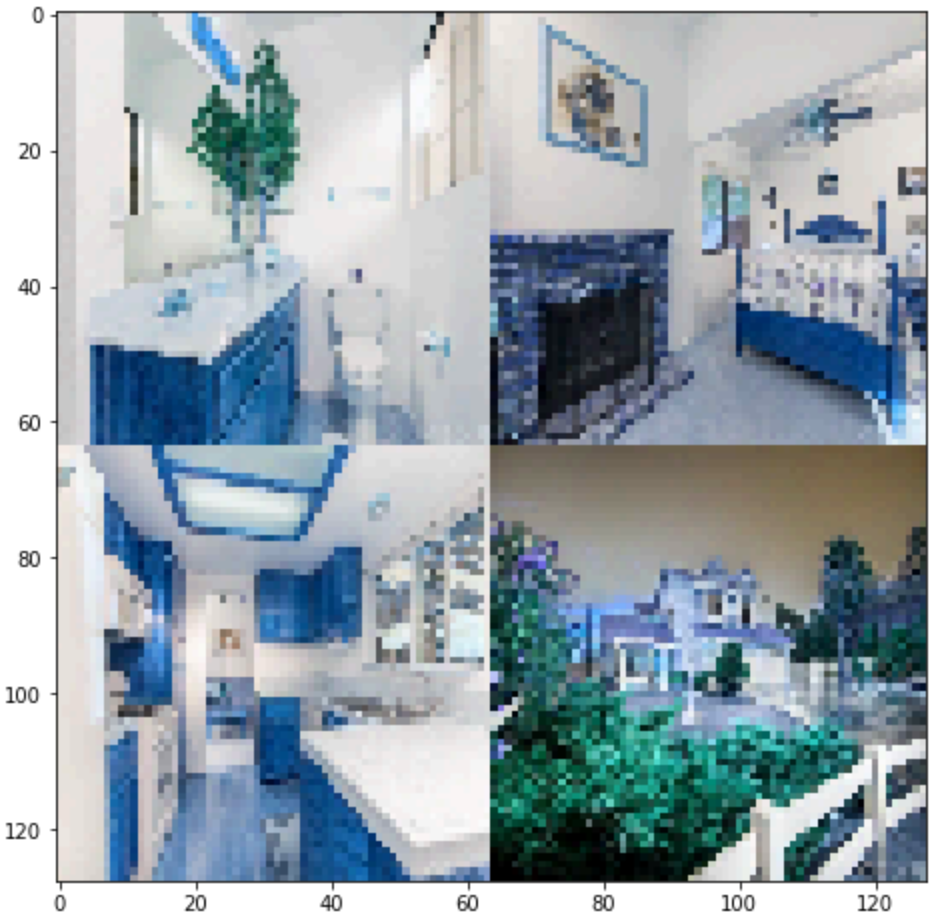

0
20
40
60
80
100
120
0
20
40
60
80
100
120


0
20
40
60
80
100
120
0
20
40
60
80
100
120

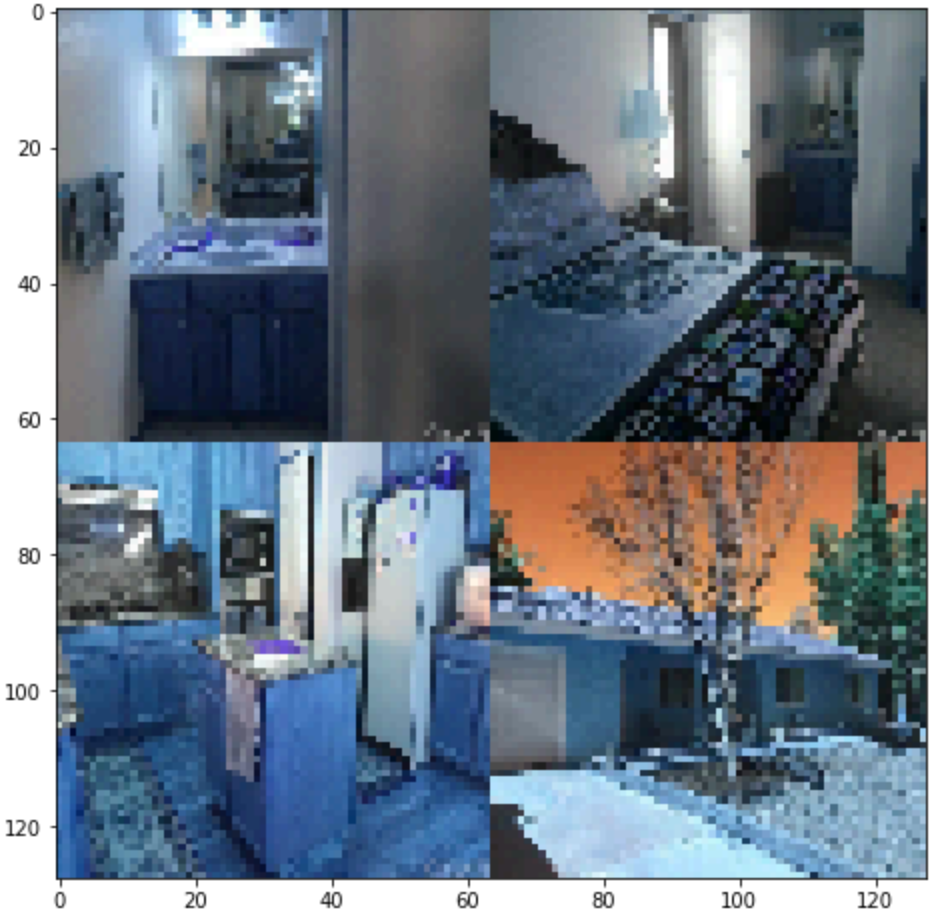

0
20
40
60
80
100
120
0
20
40
60
80
100
120

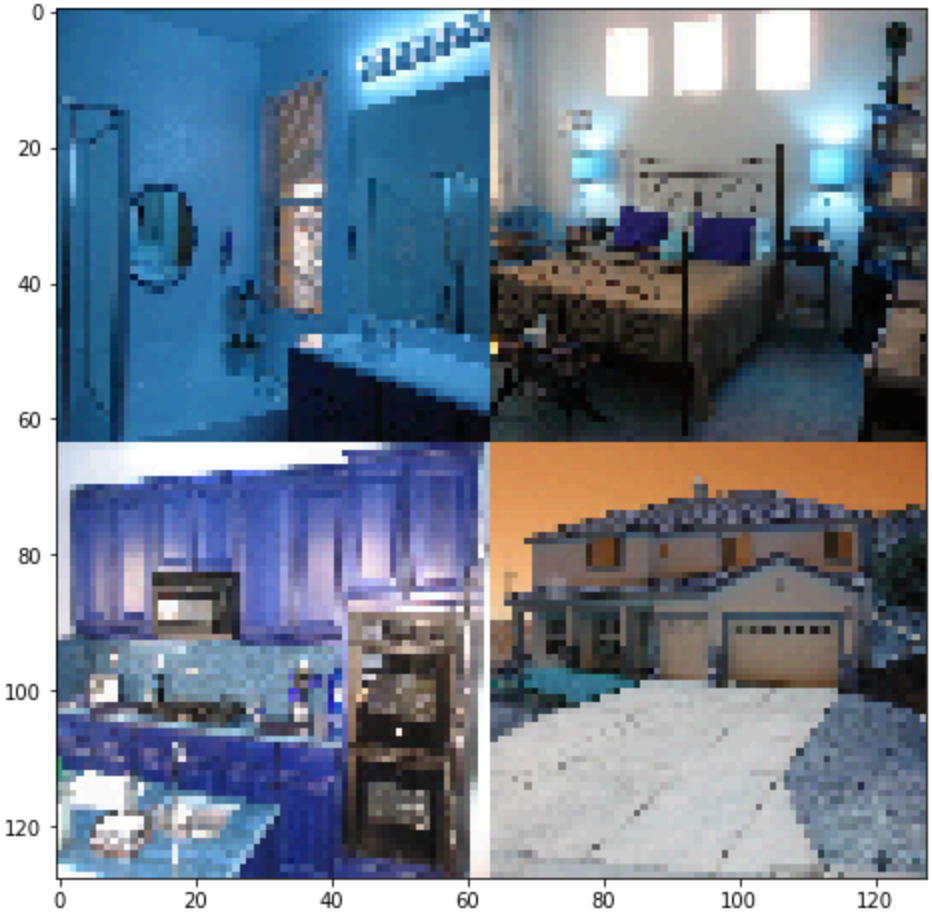

0
20
40
60
80
100
120
0
20
40
60
80
100
120

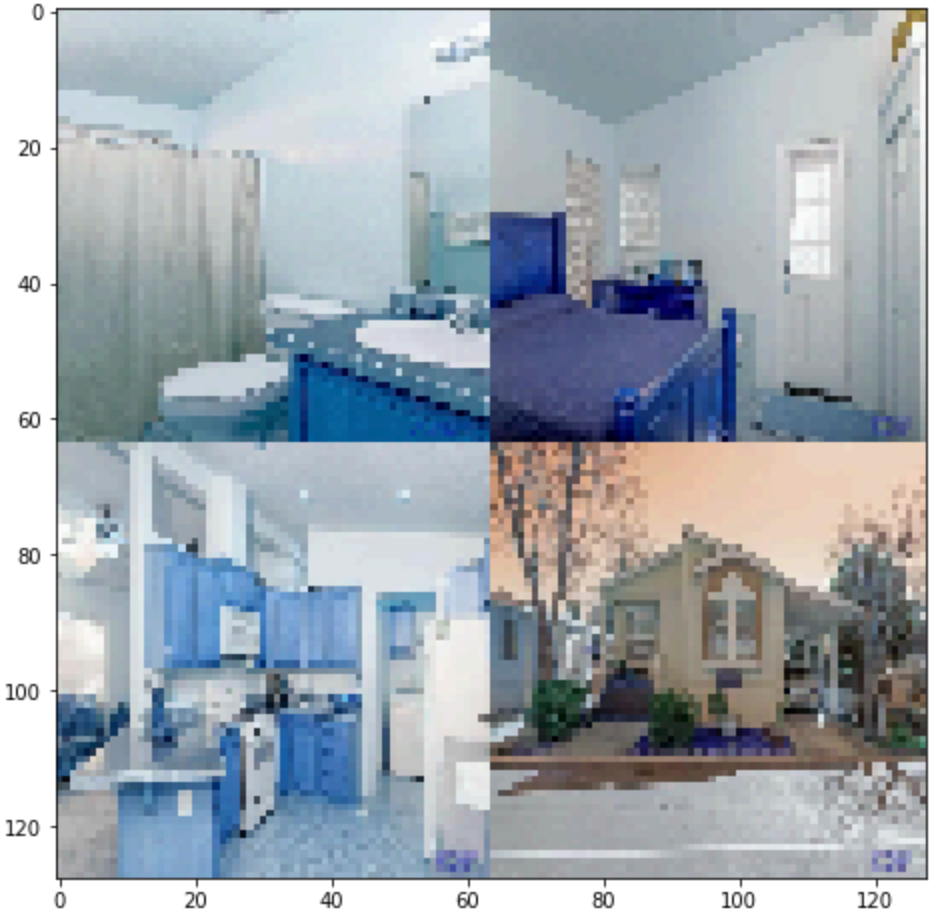

0
20
40
60
80
100
120
0
20
40
60
80
100
120


0
20
40
60
80
100
120
0
20
40
60
80
100
120


0
20
40
60
80
100
120
0
20
40
60
80
100
120

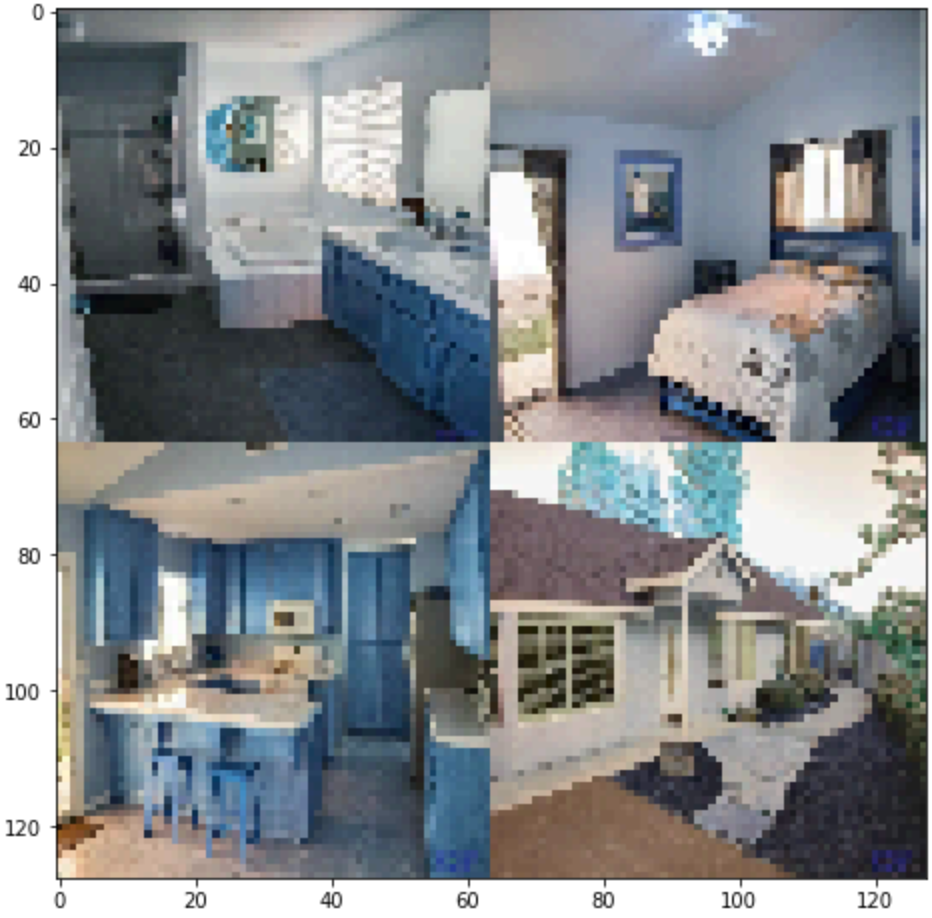

0
20
40
60
80
100
120
0
20
40
60
80
100
120

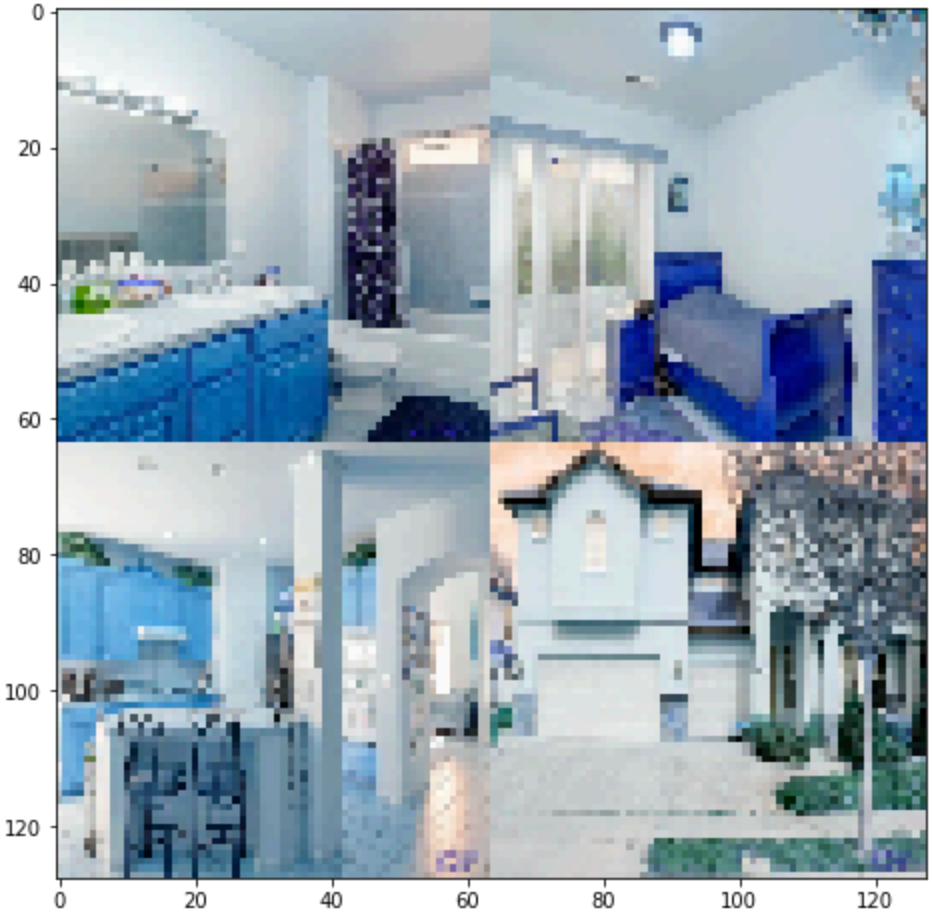

0
20
40
60
80
100
120
0
20
40
60
80
100
120

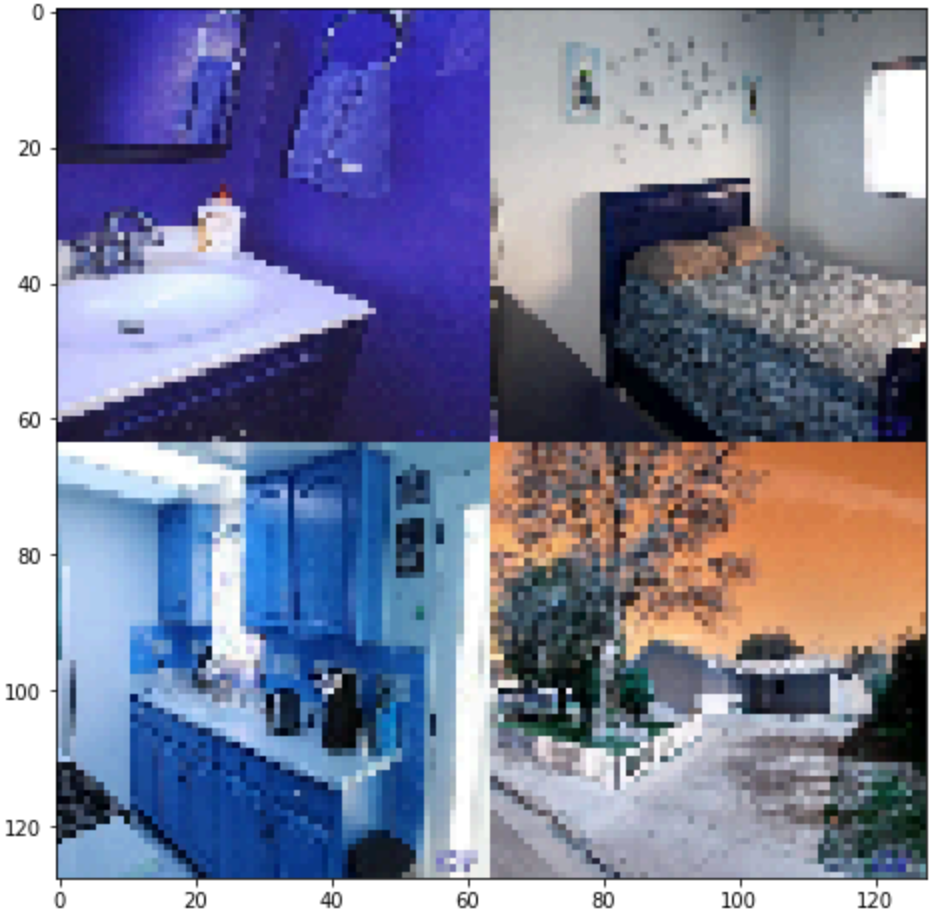

0
20
40
60
80
100
120
0
20
40
60
80
100
120

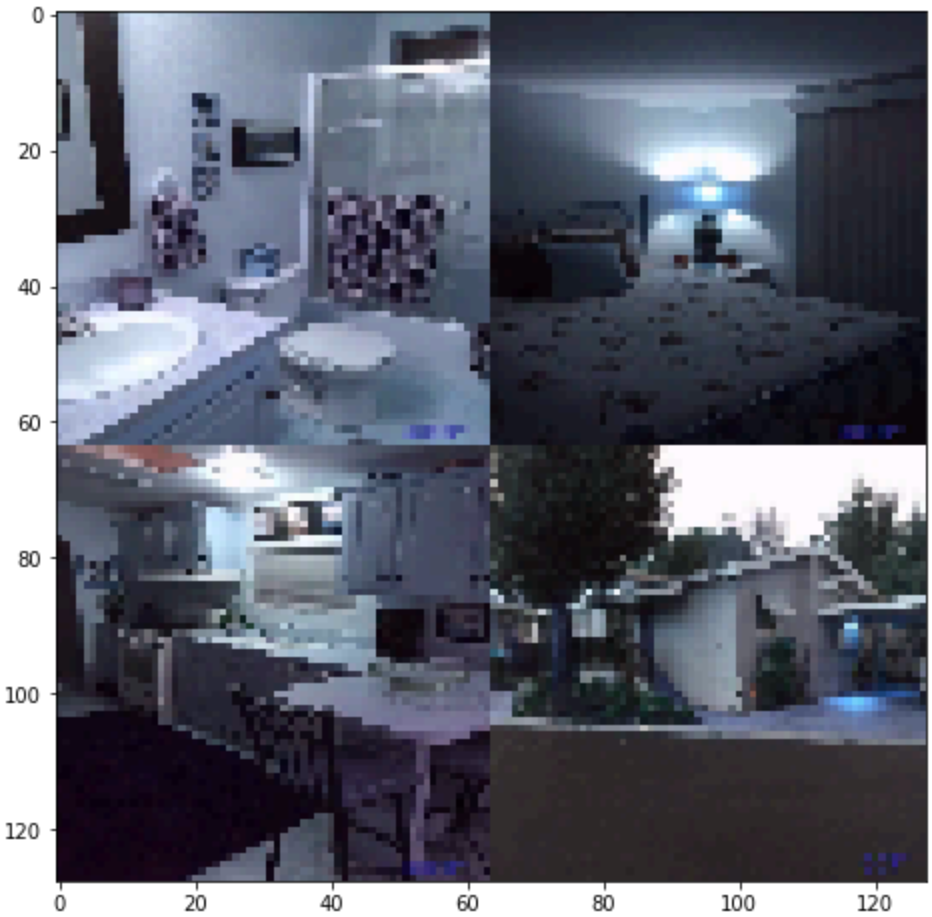

0
20
40
60
80
100
120
0
20
40
60
80
100
120


0
20
40
60
80
100
120
0
20
40
60
80
100
120


0
20
40
60
80
100
120
0
20
40
60
80
100
120


0
20
40
60
80
100
120
0
20
40
60
80
100
120

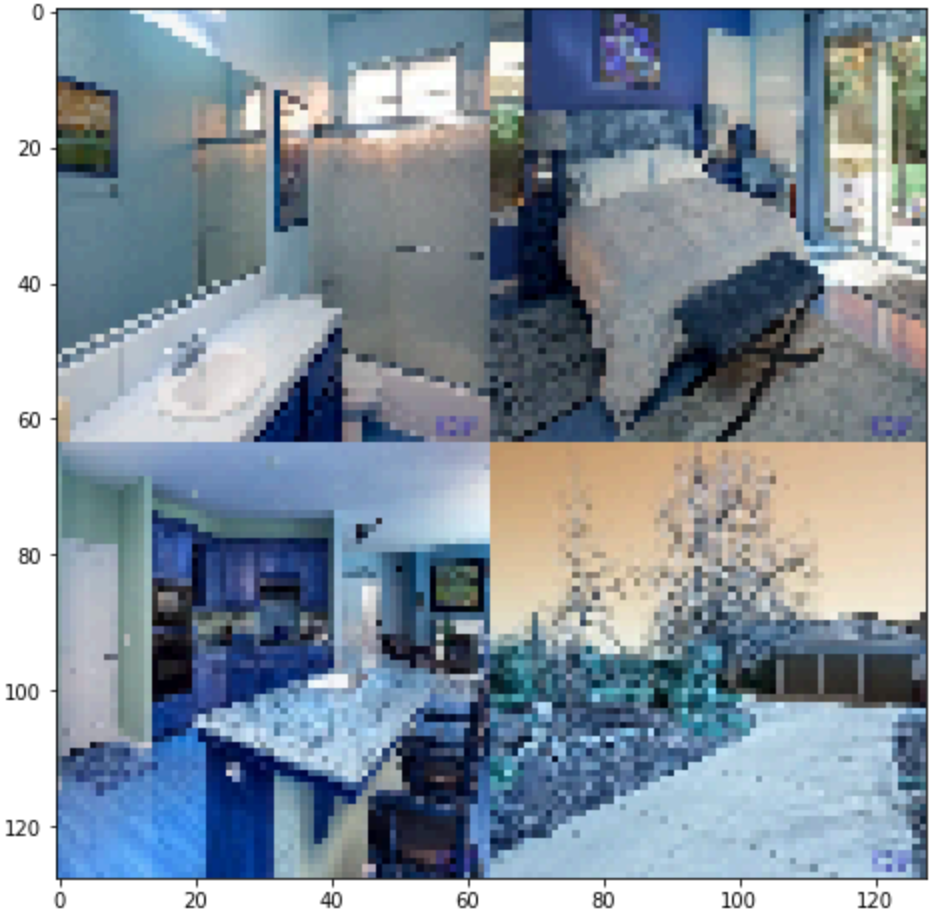

0
20
40
60
80
100
120
0
20
40
60
80
100
120


0
20
40
60
80
100
120
0
20
40
60
80
100
120

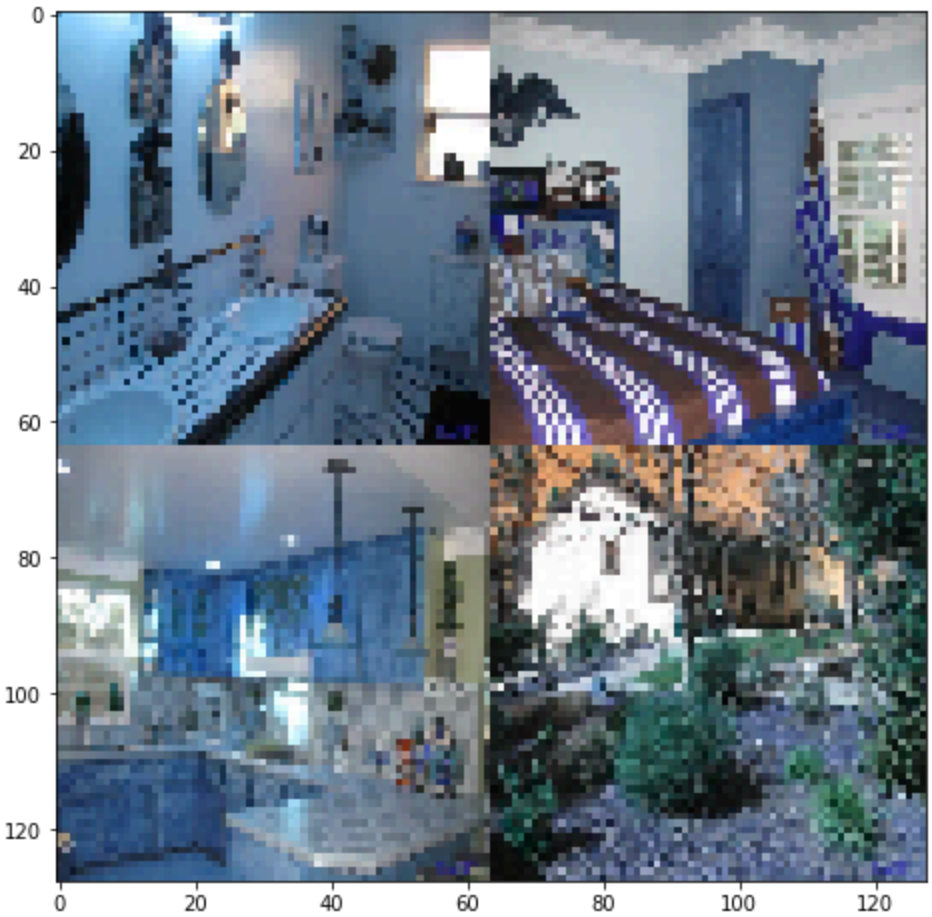

0
20
40
60
80
100
120
0
20
40
60
80
100
120


0
20
40
60
80
100
120
0
20
40
60
80
100
120

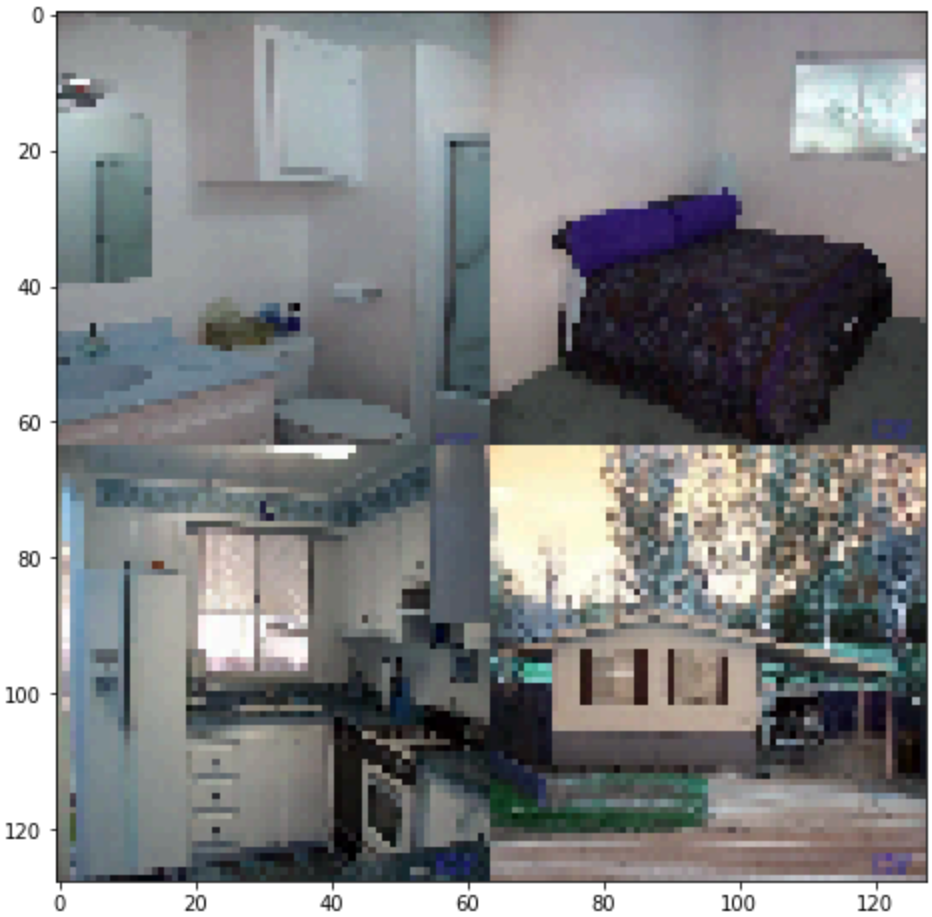

0
20
40
60
80
100
120
0
20
40
60
80
100
120


0
20
40
60
80
100
120
0
20
40
60
80
100
120


0
20
40
60
80
100
120
0
20
40
60
80
100
120


0
20
40
60
80
100
120
0
20
40
60
80
100
120

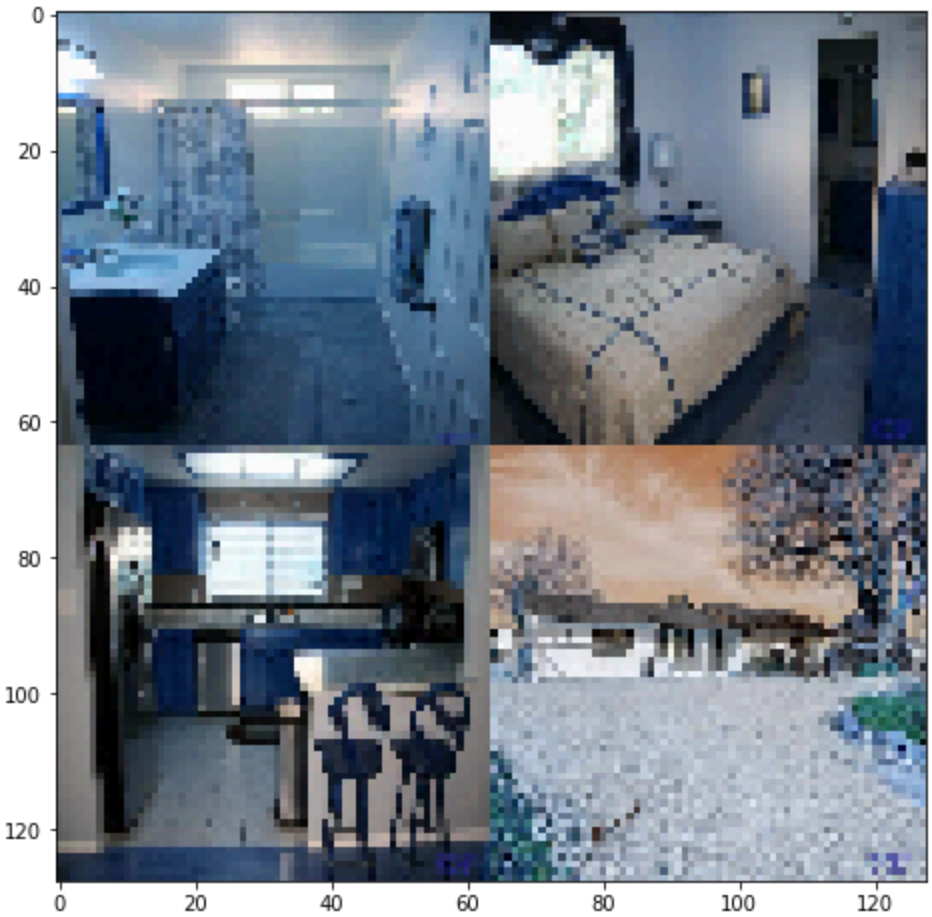

0
20
40
60
80
100
120
0
20
40
60
80
100
120


0
20
40
60
80
100
120
0
20
40
60
80
100
120

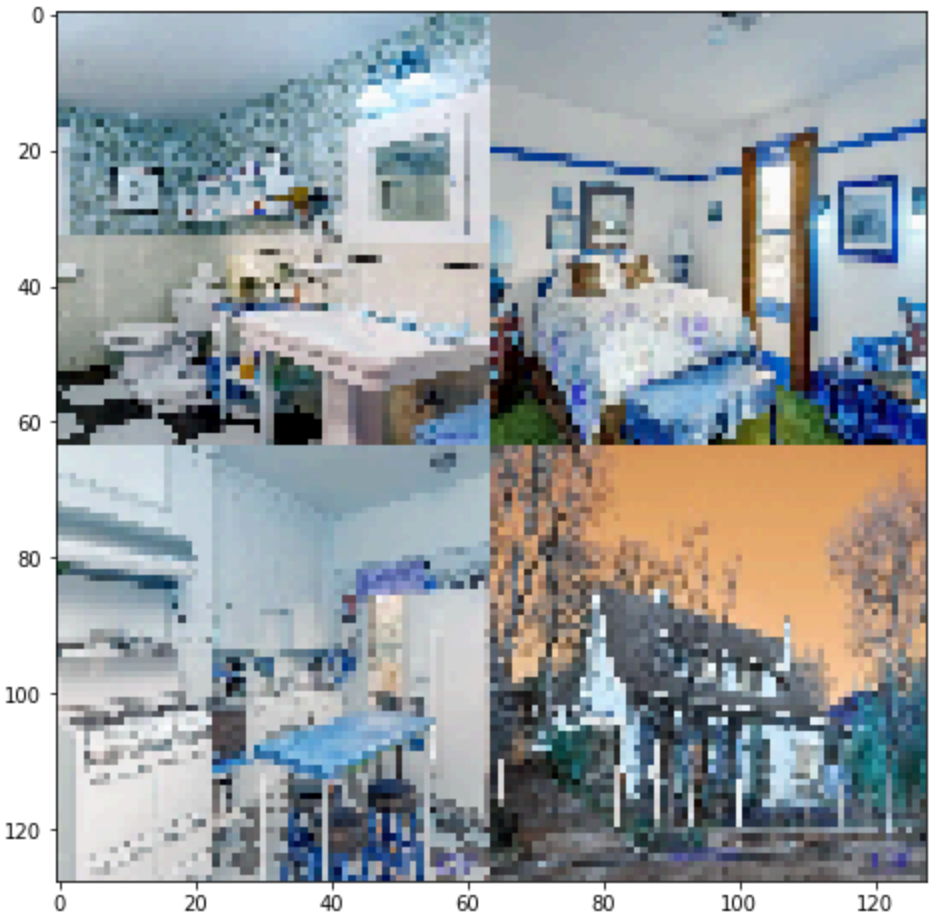

0
20
40
60
80
100
120
0
20
40
60
80
100
120


0
20
40
60
80
100
120
0
20
40
60
80
100
120


0
20
40
60
80
100
120
0
20
40
60
80
100
120

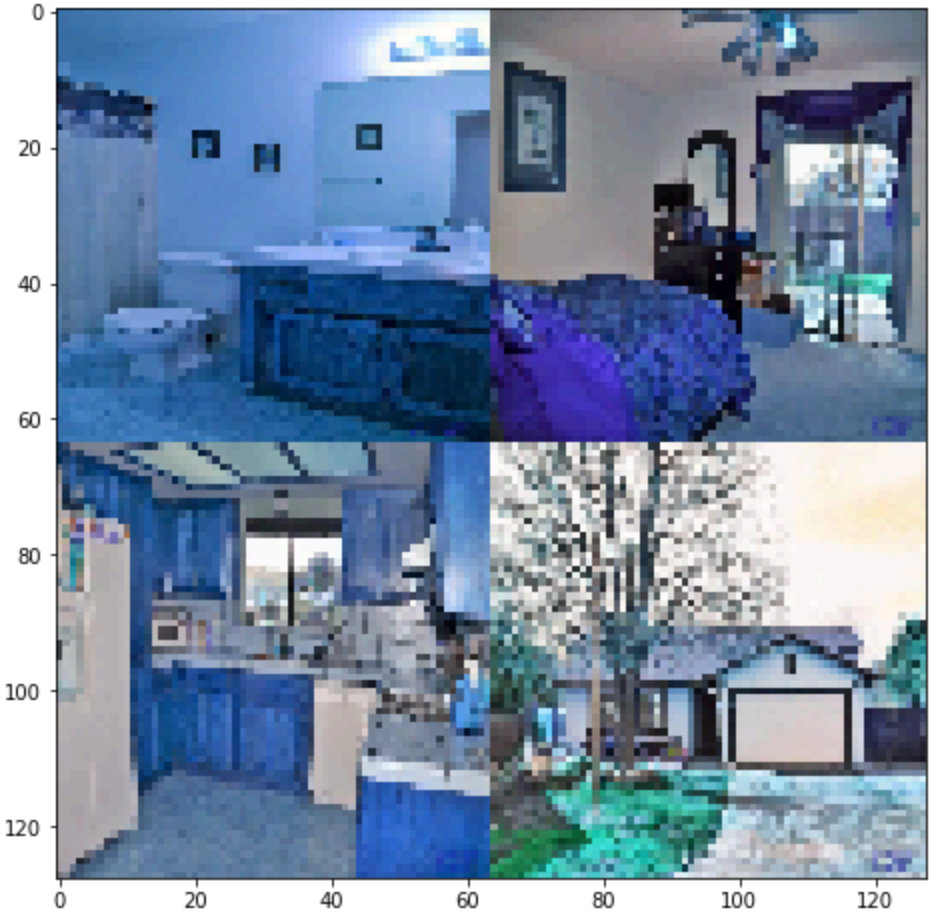

0
20
40
60
80
100
120
0
20
40
60
80
100
120

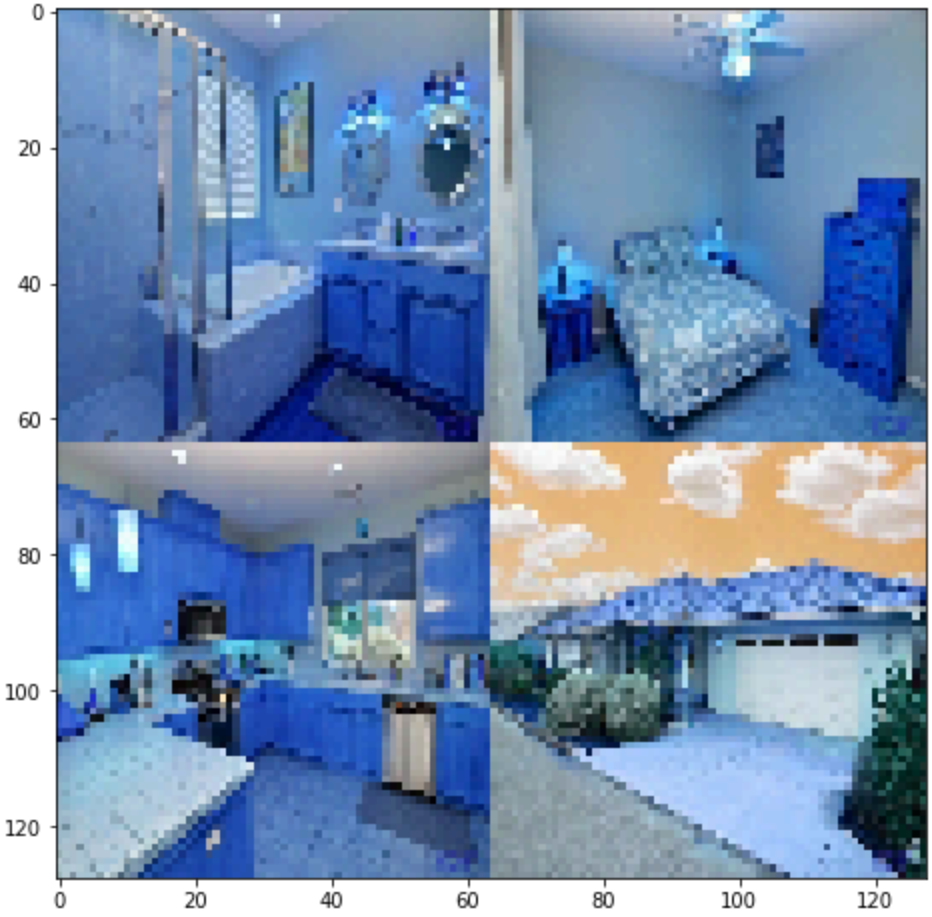

0
20
40
60
80
100
120
0
20
40
60
80
100
120

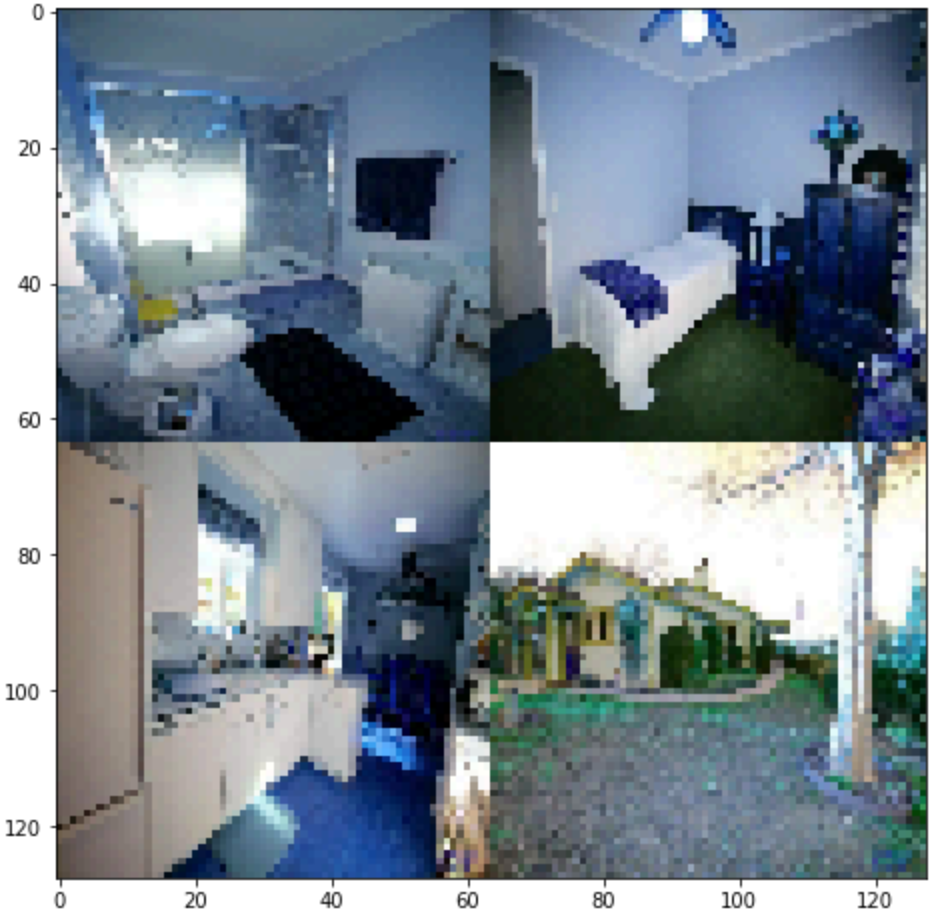

0
20
40
60
80
100
120
0
20
40
60
80
100
120

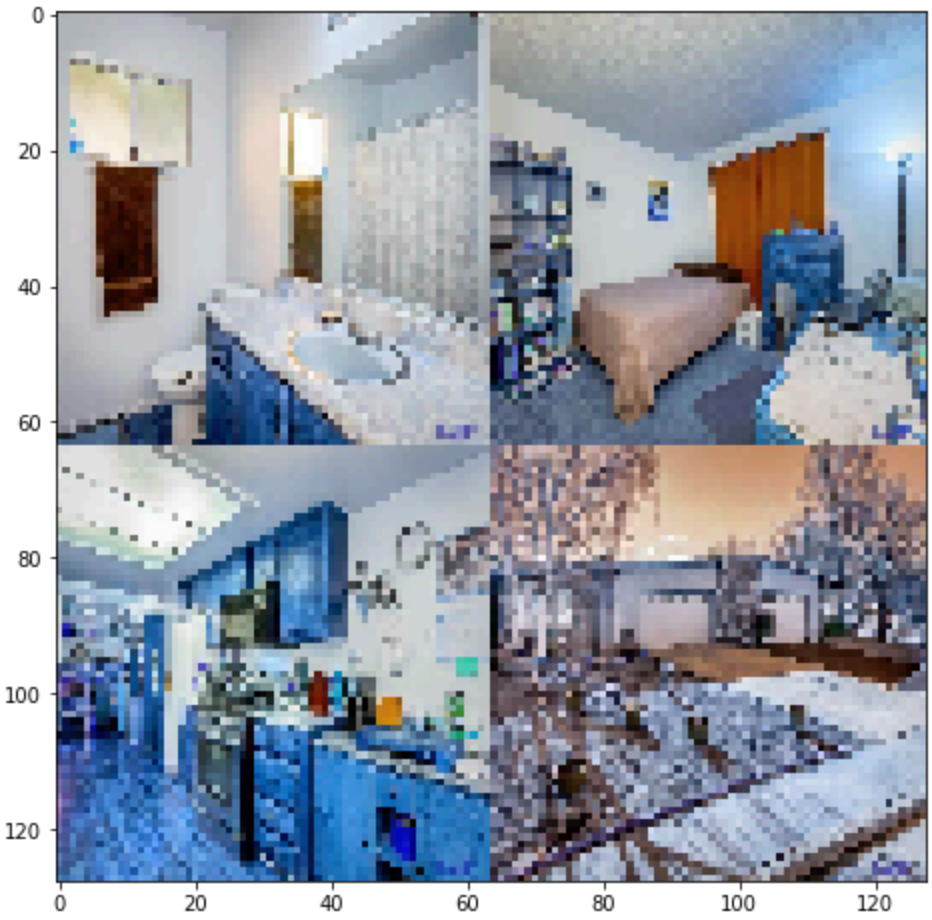

0
20
40
60
80
100
120
0
20
40
60
80
100
120

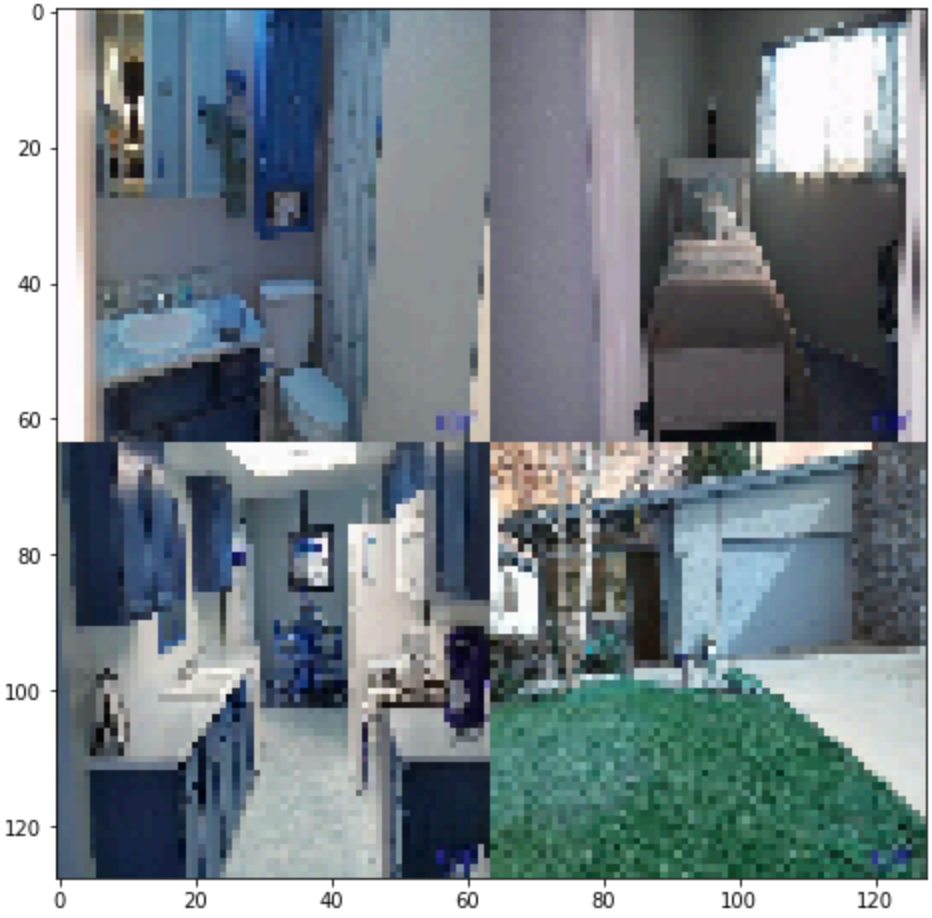

0
20
40
60
80
100
120
0
20
40
60
80
100
120


0
20
40
60
80
100
120
0
20
40
60
80
100
120

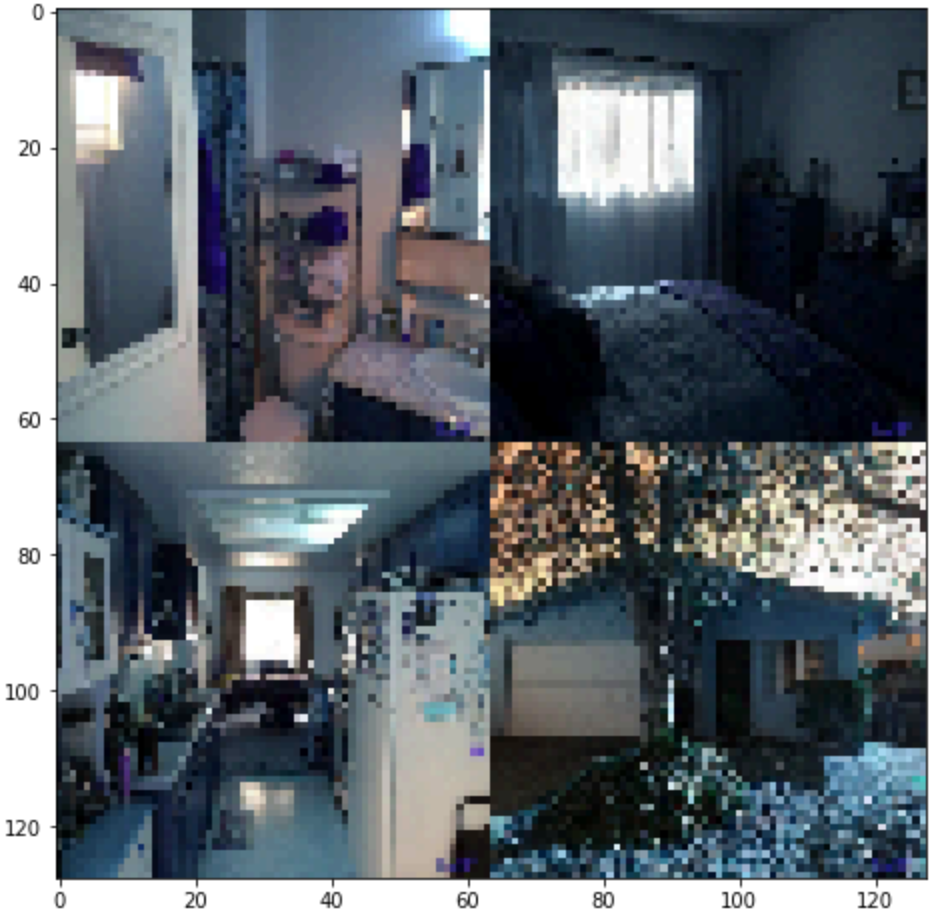

0
20
40
60
80
100
120
0
20
40
60
80
100
120


0
20
40
60
80
100
120
0
20
40
60
80
100
120

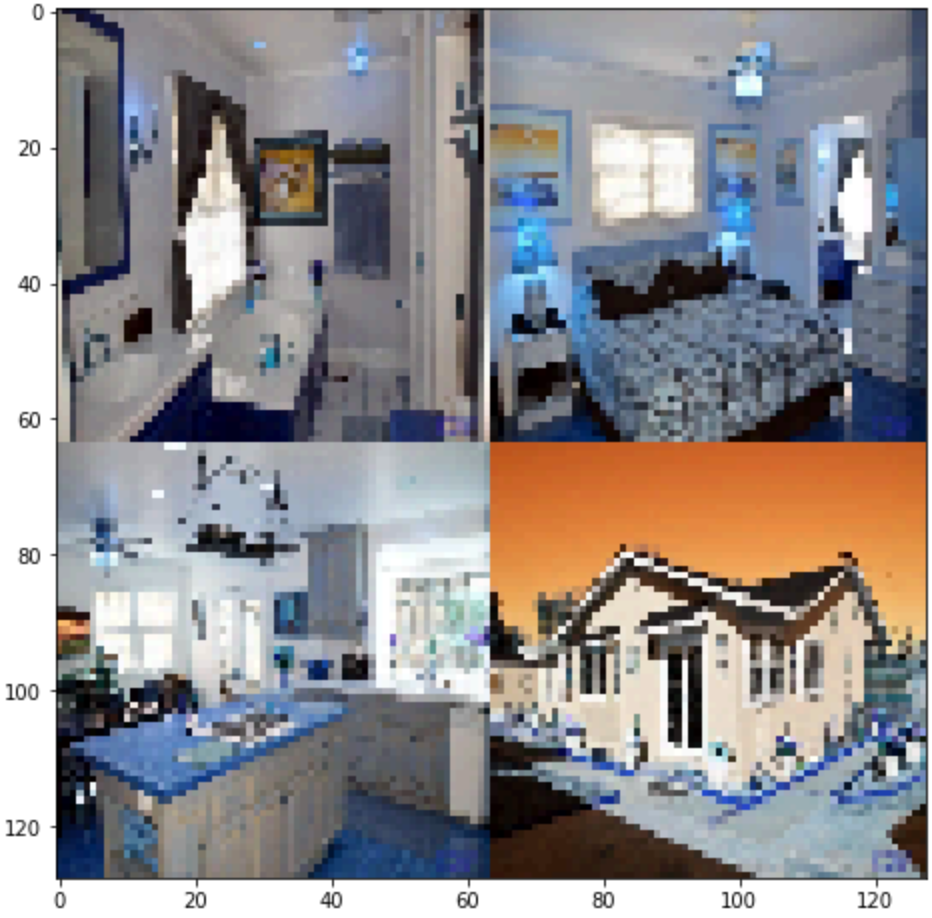

0
20
40
60
80
100
120
0
20
40
60
80
100
120


0
20
40
60
80
100
120
0
20
40
60
80
100
120

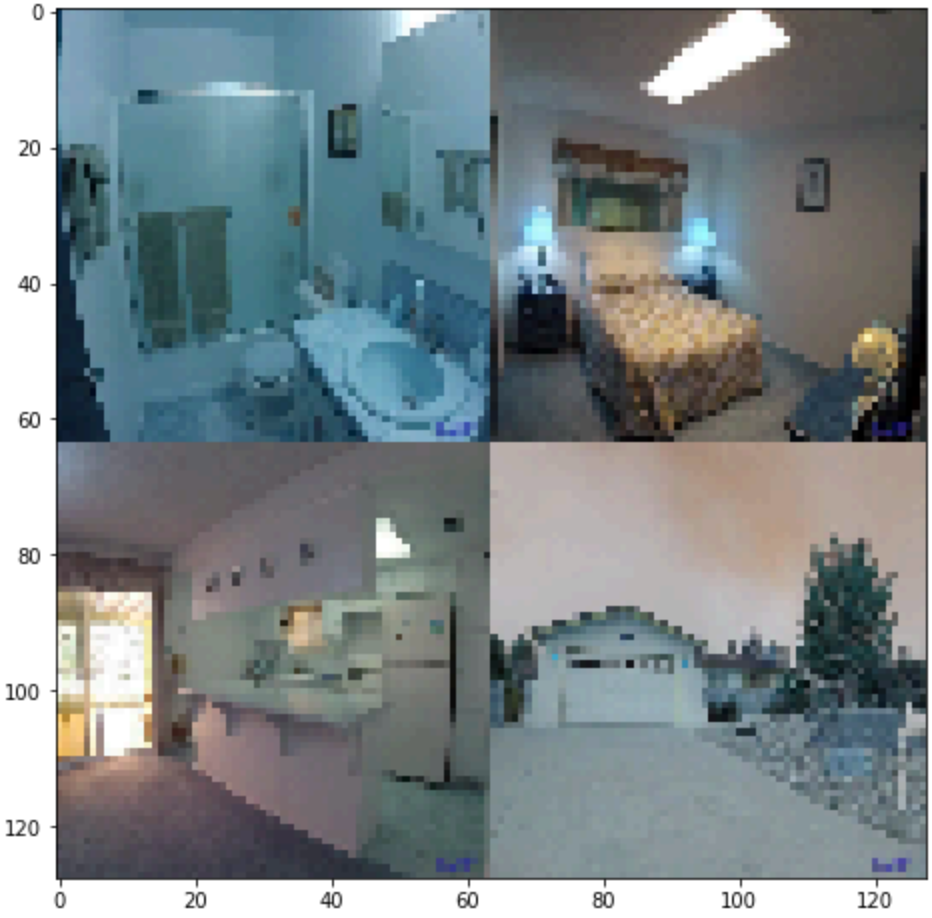

0
20
40
60
80
100
120
0
20
40
60
80
100
120

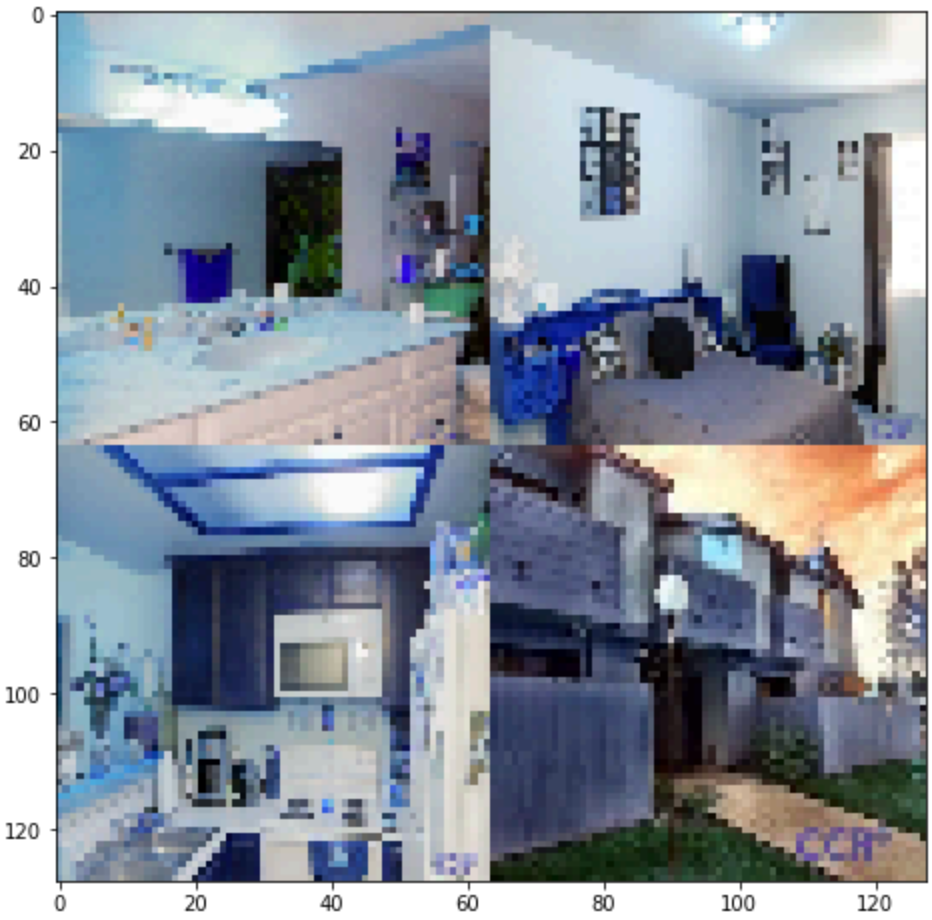

0
20
40
60
80
100
120
0
20
40
60
80
100
120

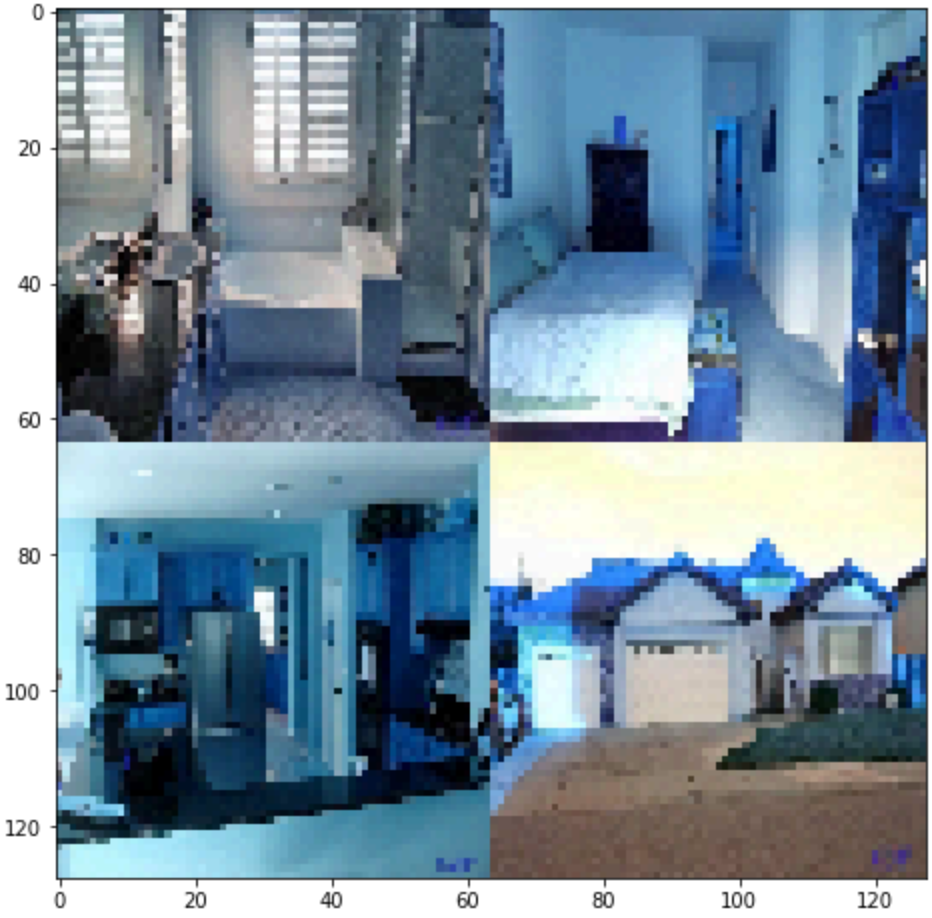

0
20
40
60
80
100
120
0
20
40
60
80
100
120

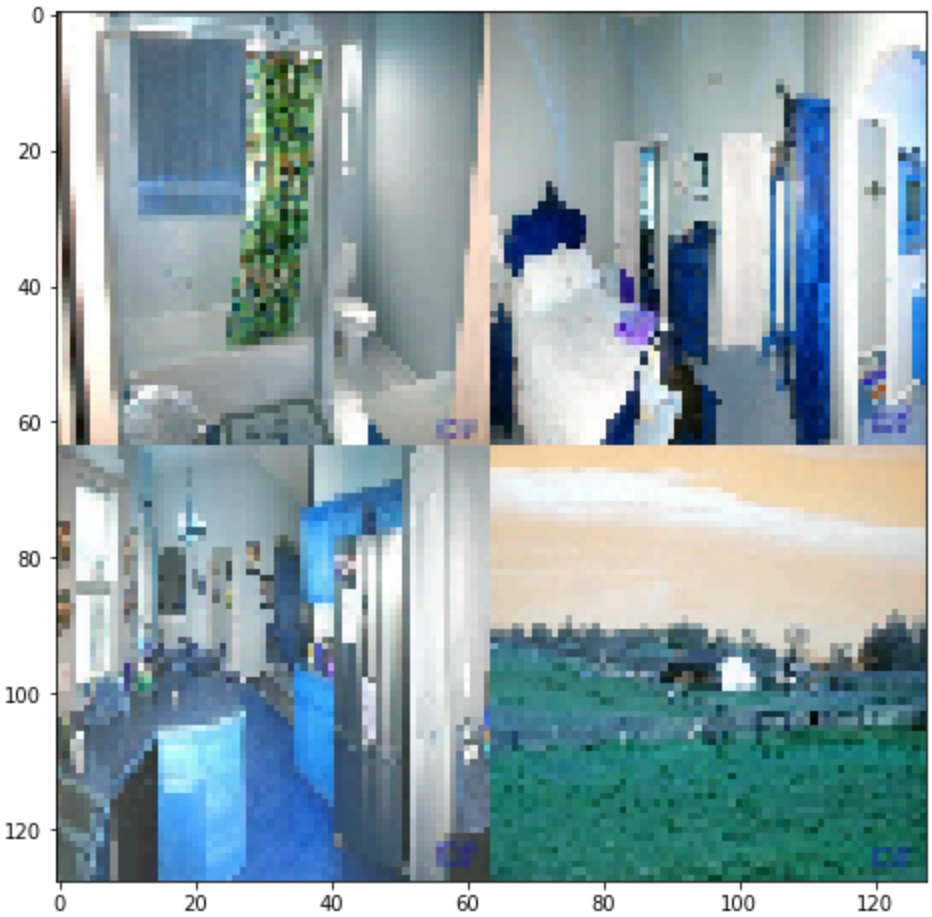

0
20
40
60
80
100
120
0
20
40
60
80
100
120

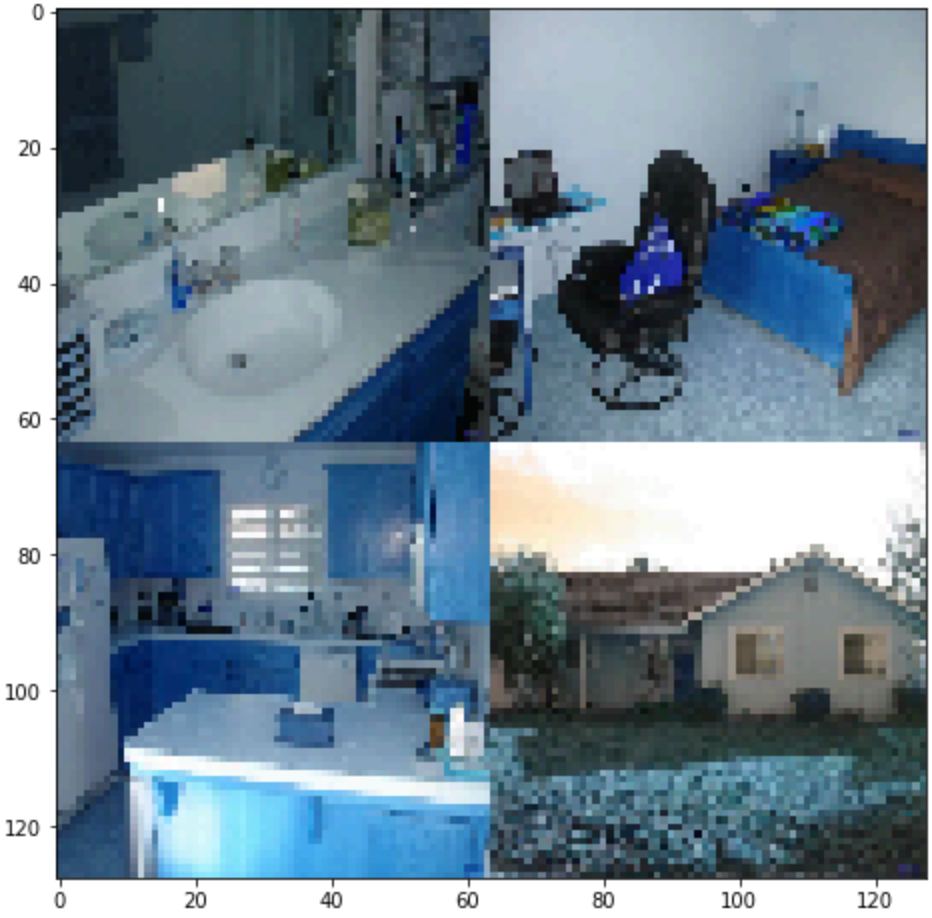

0
20
40
60
80
100
120
0
20
40
60
80
100
120

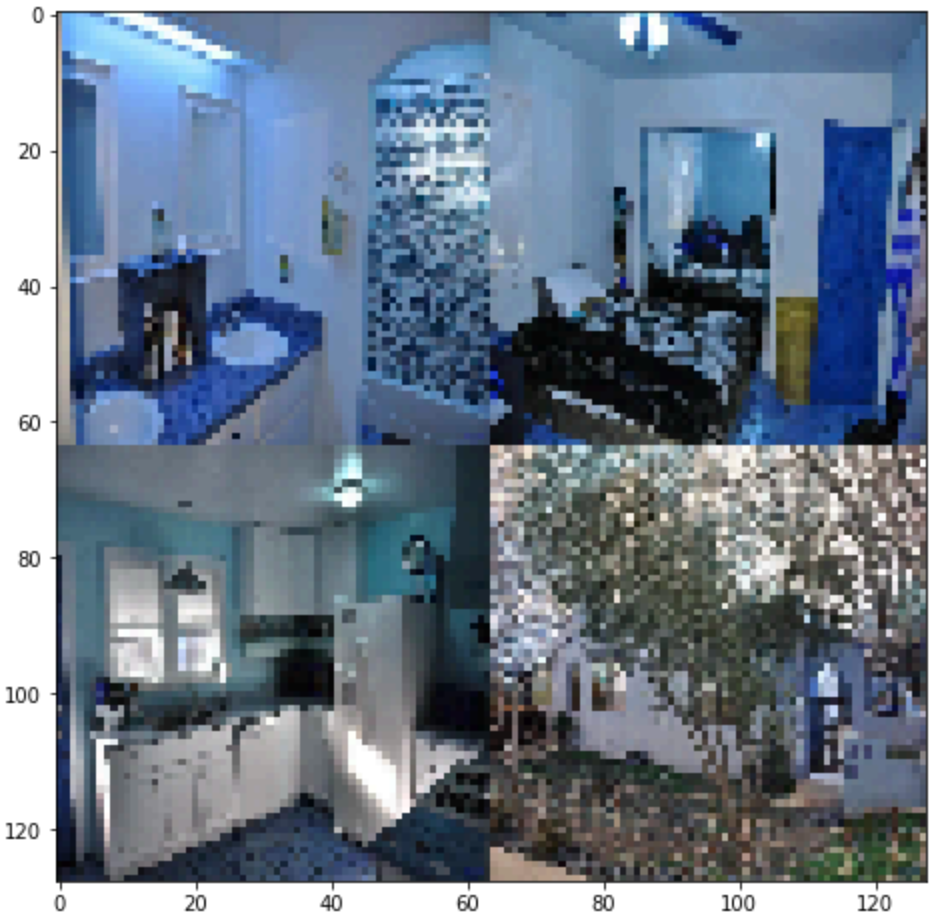

0
20
40
60
80
100
120
0
20
40
60
80
100
120


0
20
40
60
80
100
120
0
20
40
60
80
100
120

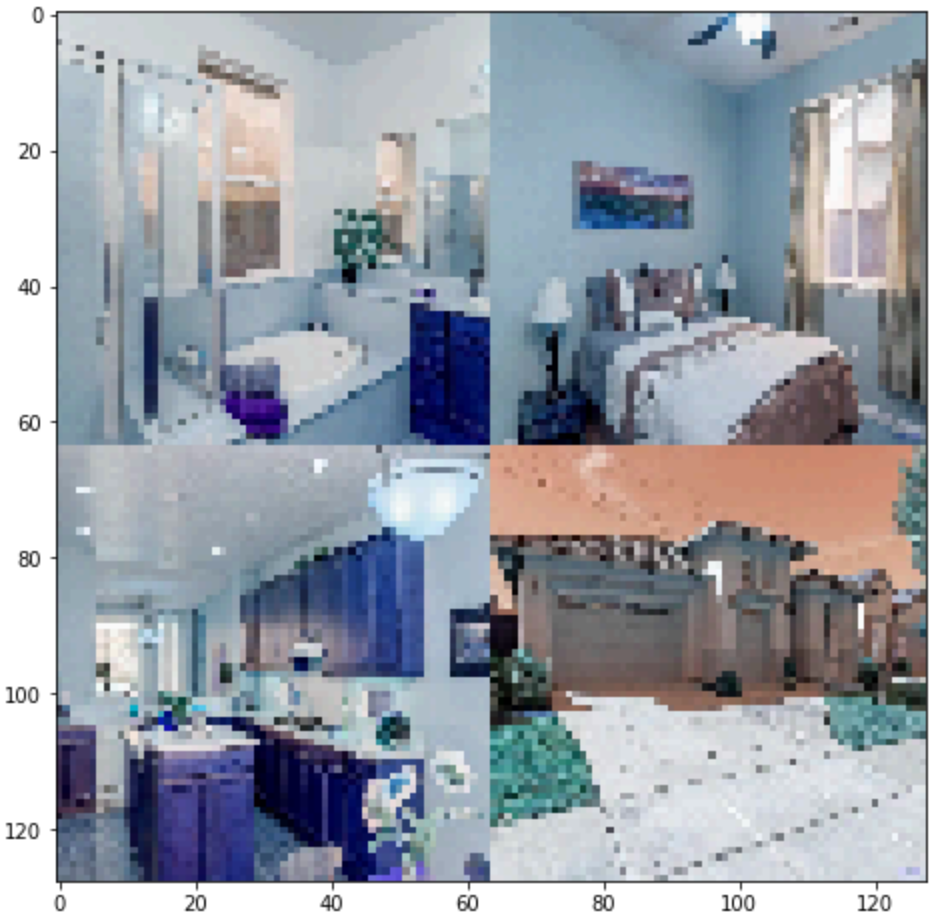

0
20
40
60
80
100
120
0
20
40
60
80
100
120

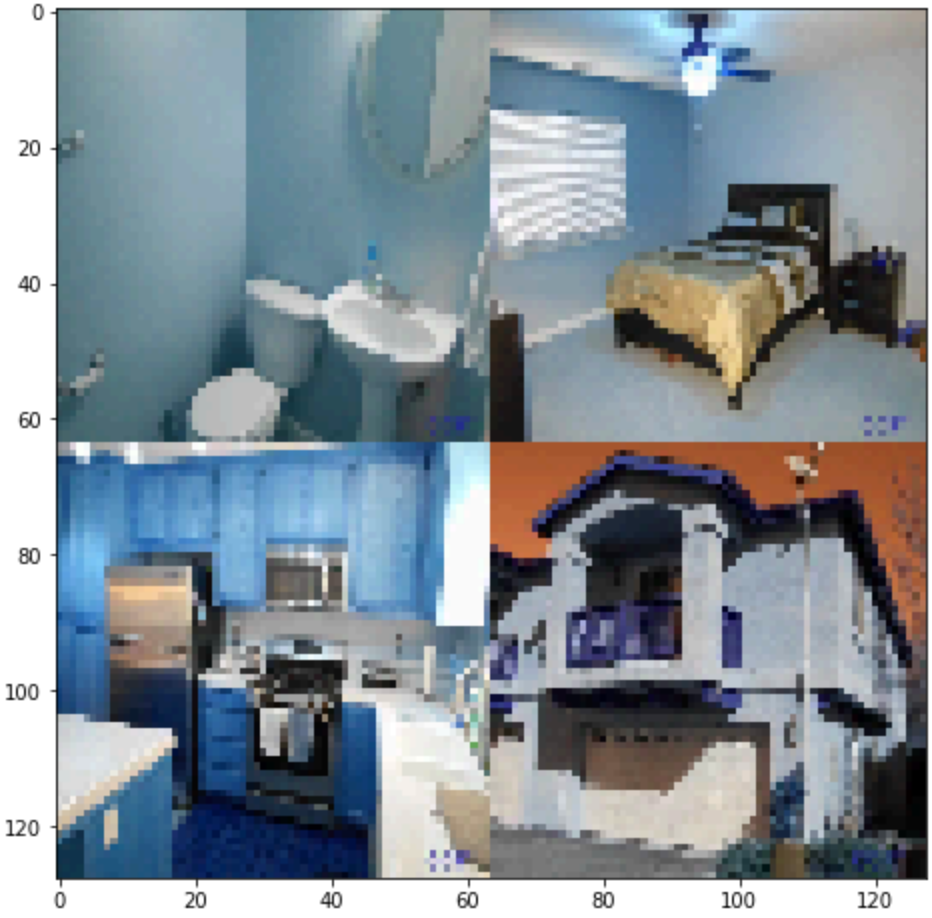

0
20
40
60
80
100
120
0
20
40
60
80
100
120

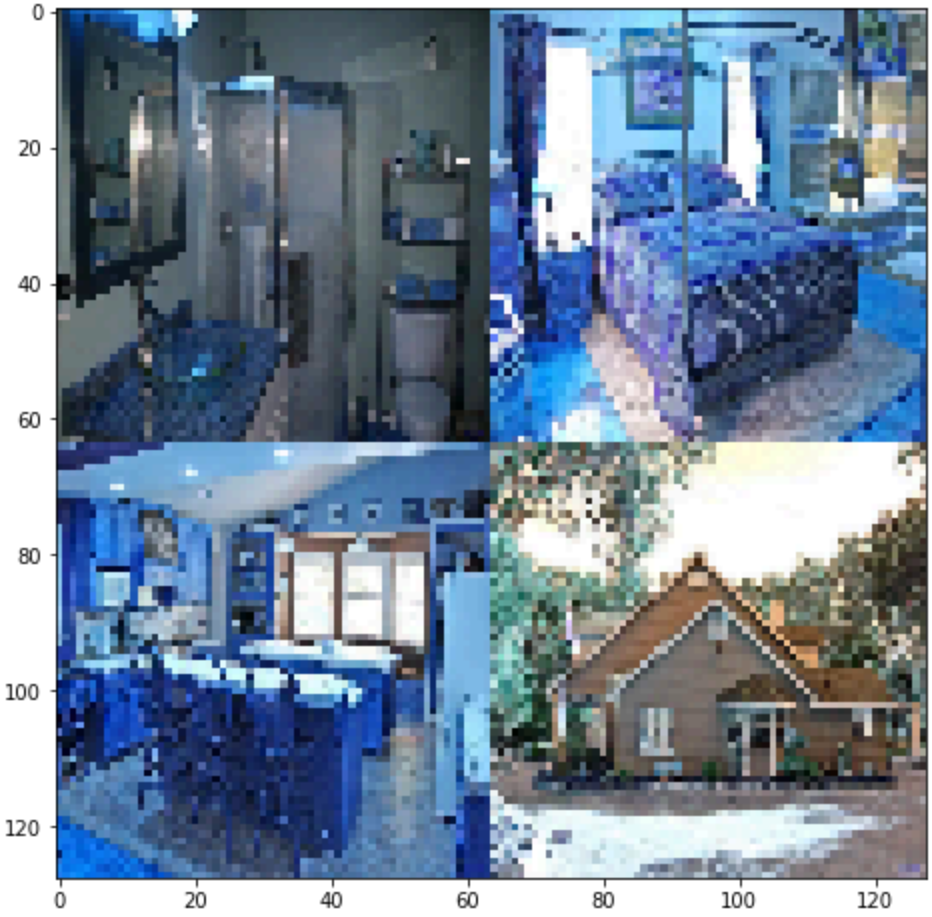

0
20
40
60
80
100
120
0
20
40
60
80
100
120

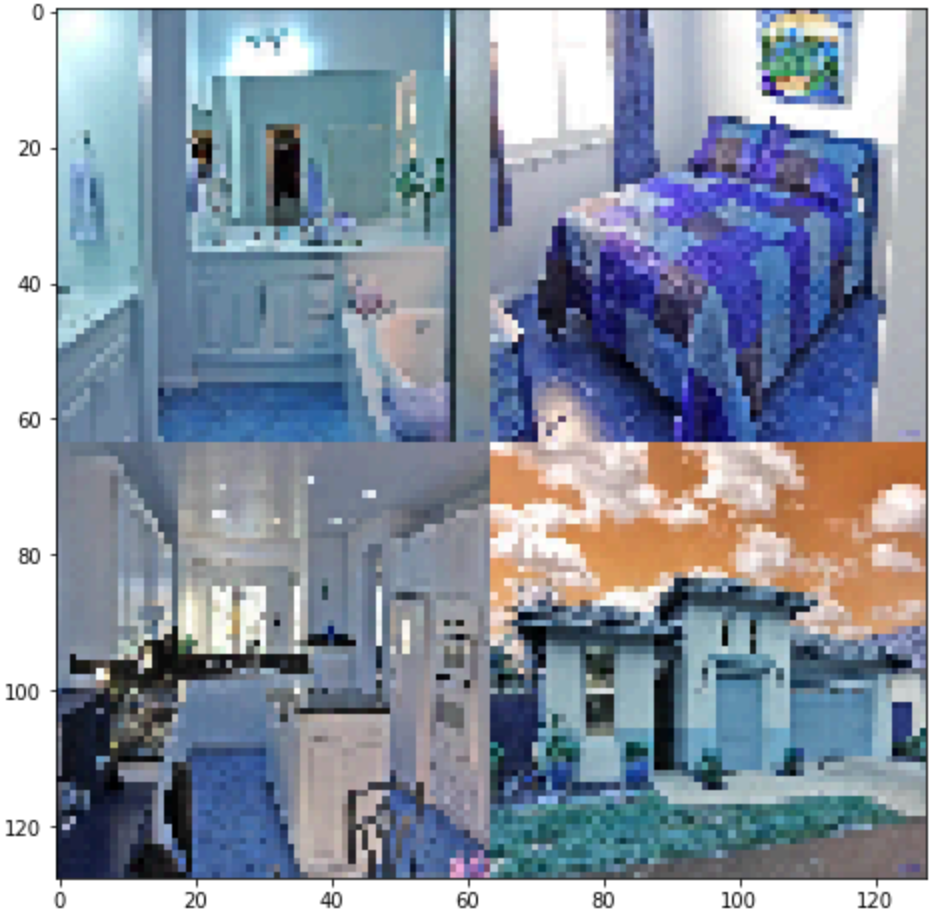

0
20
40
60
80
100
120
0
20
40
60
80
100
120

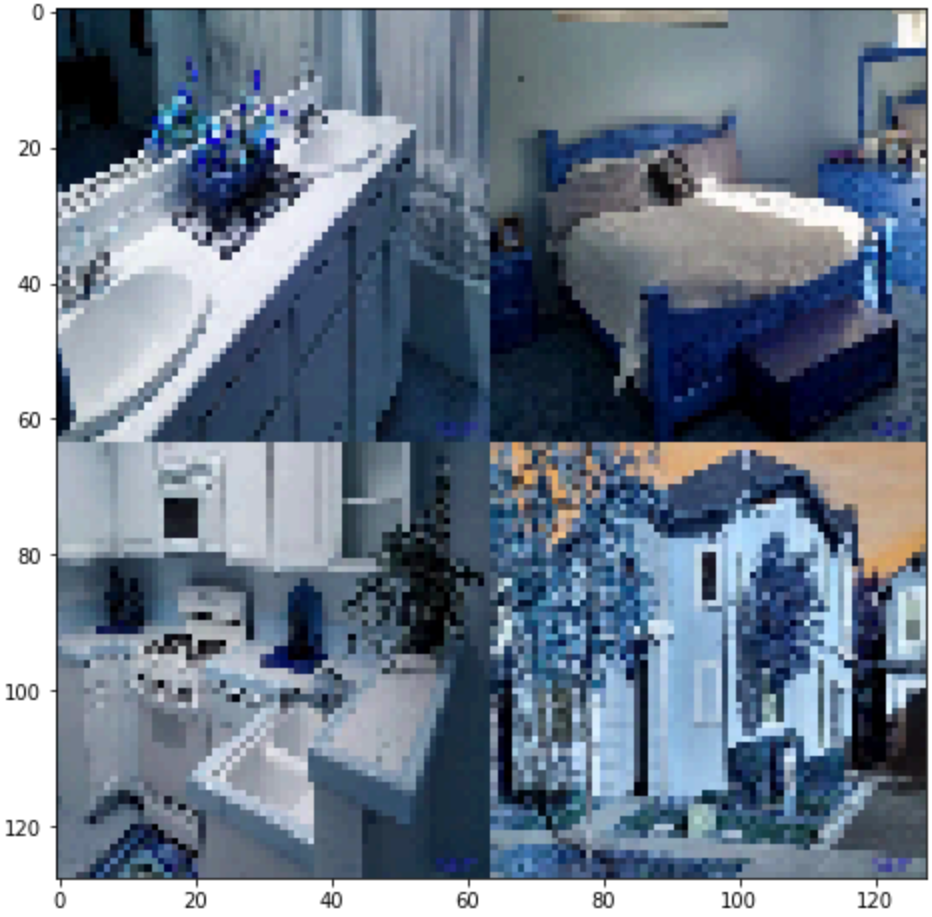

0
20
40
60
80
100
120
0
20
40
60
80
100
120


0
20
40
60
80
100
120
0
20
40
60
80
100
120

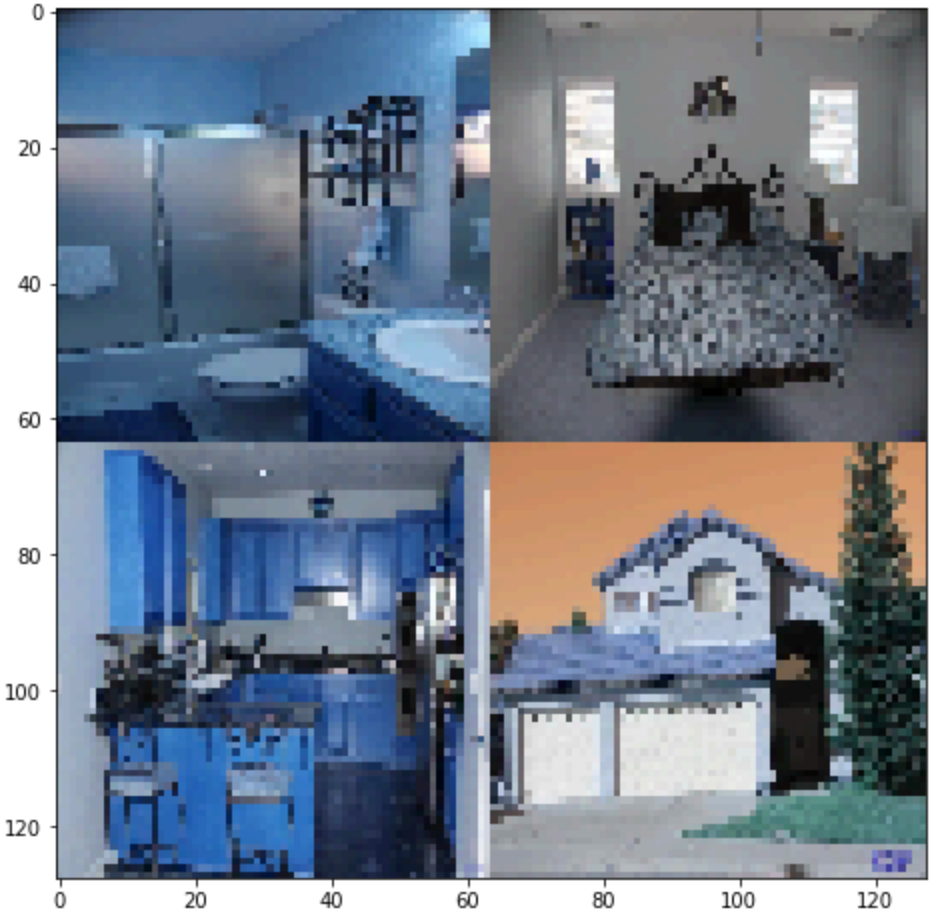

0
20
40
60
80
100
120
0
20
40
60
80
100
120


0
20
40
60
80
100
120
0
20
40
60
80
100
120

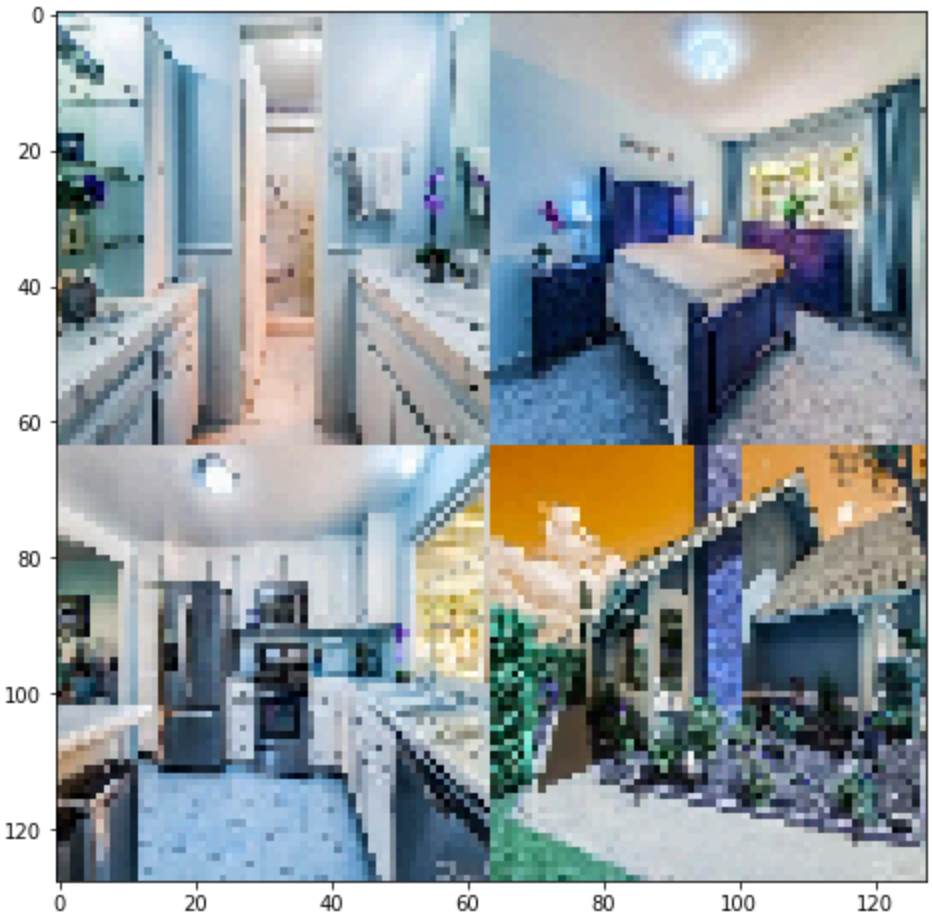

0
20
40
60
80
100
120
0
20
40
60
80
100
120

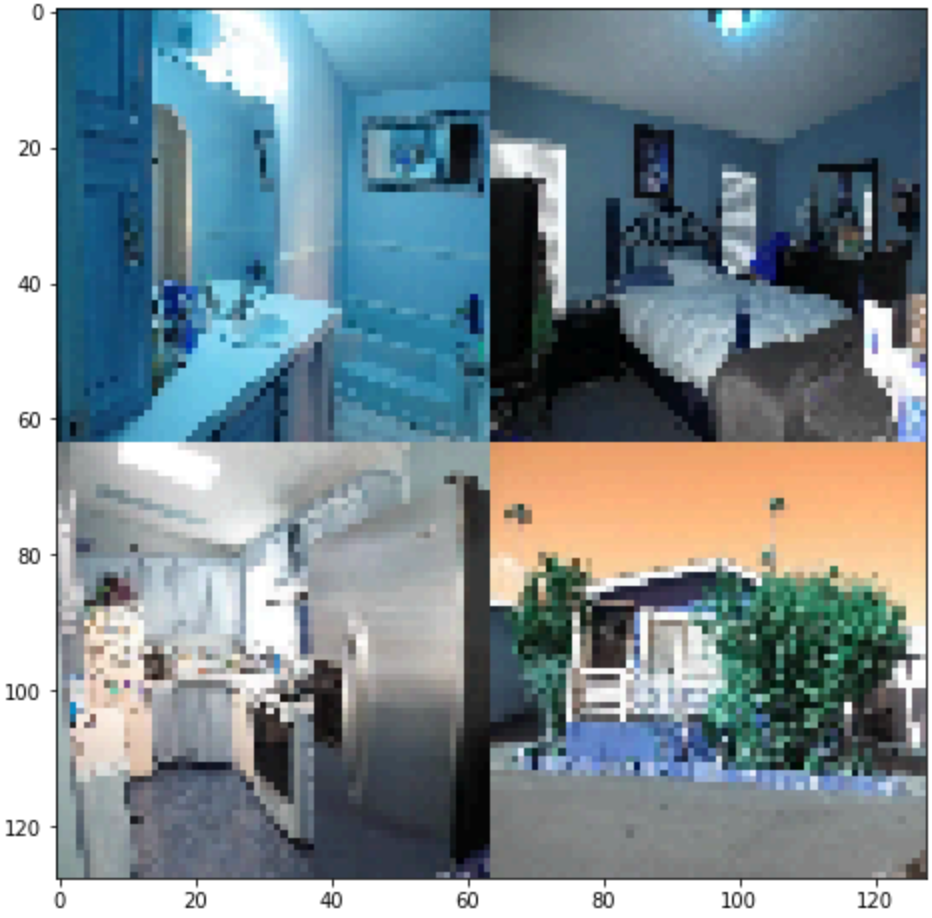

0
20
40
60
80
100
120
0
20
40
60
80
100
120

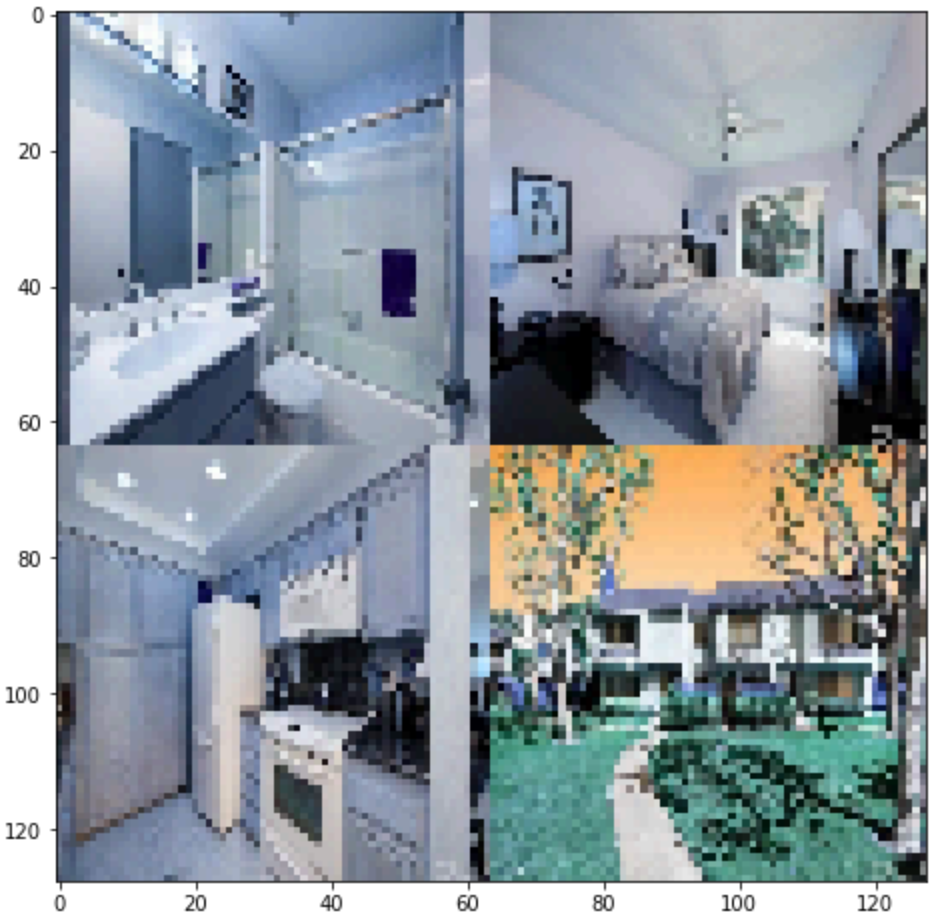

0
20
40
60
80
100
120
0
20
40
60
80
100
120

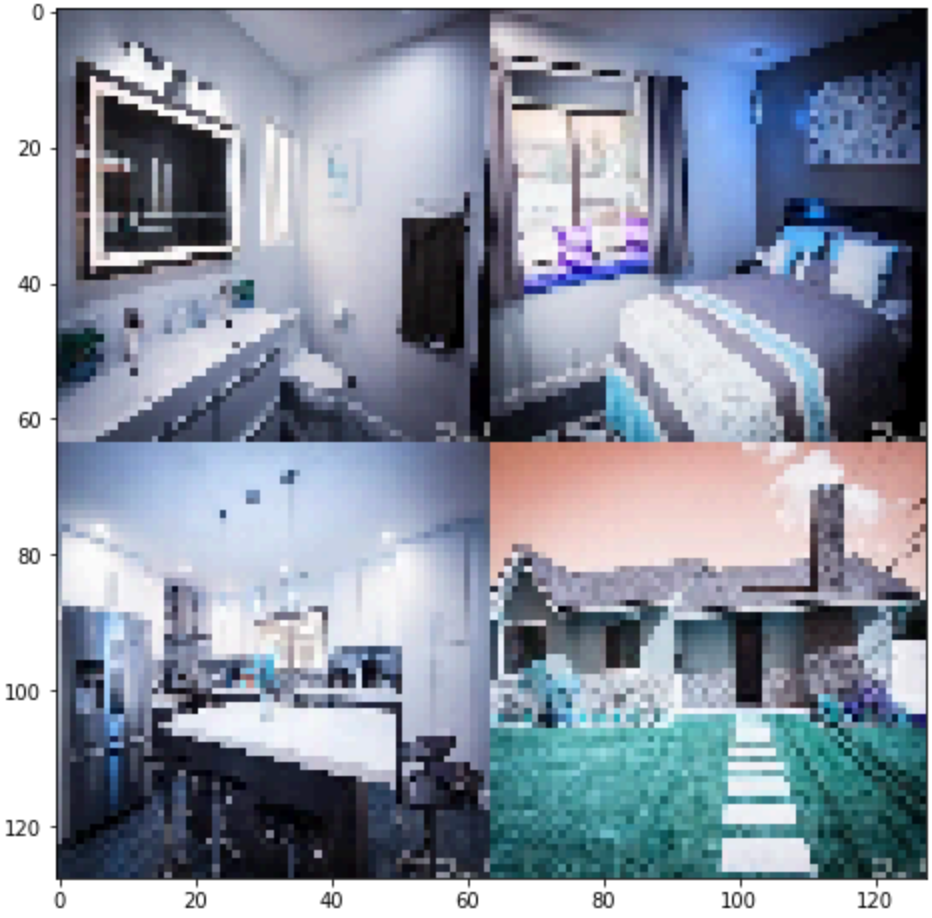

0
20
40
60
80
100
120
0
20
40
60
80
100
120


0
20
40
60
80
100
120
0
20
40
60
80
100
120


0
20
40
60
80
100
120
0
20
40
60
80
100
120


0
20
40
60
80
100
120
0
20
40
60
80
100
120

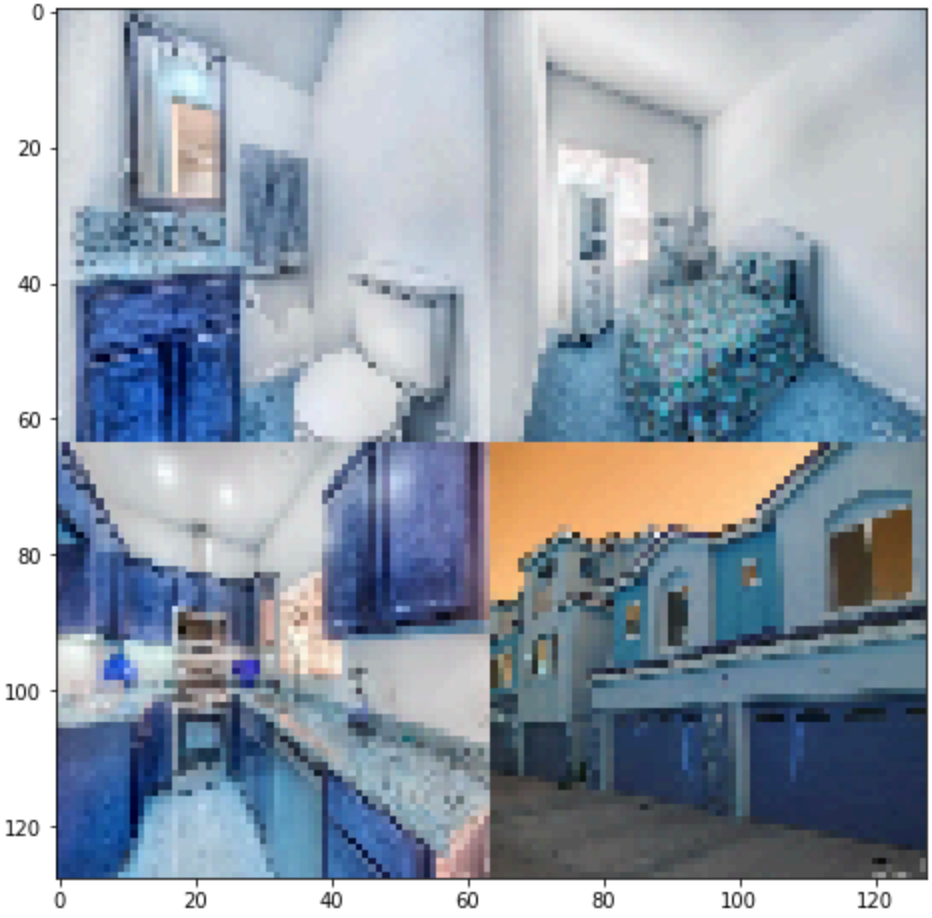

0
20
40
60
80
100
120
0
20
40
60
80
100
120


0
20
40
60
80
100
120
0
20
40
60
80
100
120


0
20
40
60
80
100
120
0
20
40
60
80
100
120

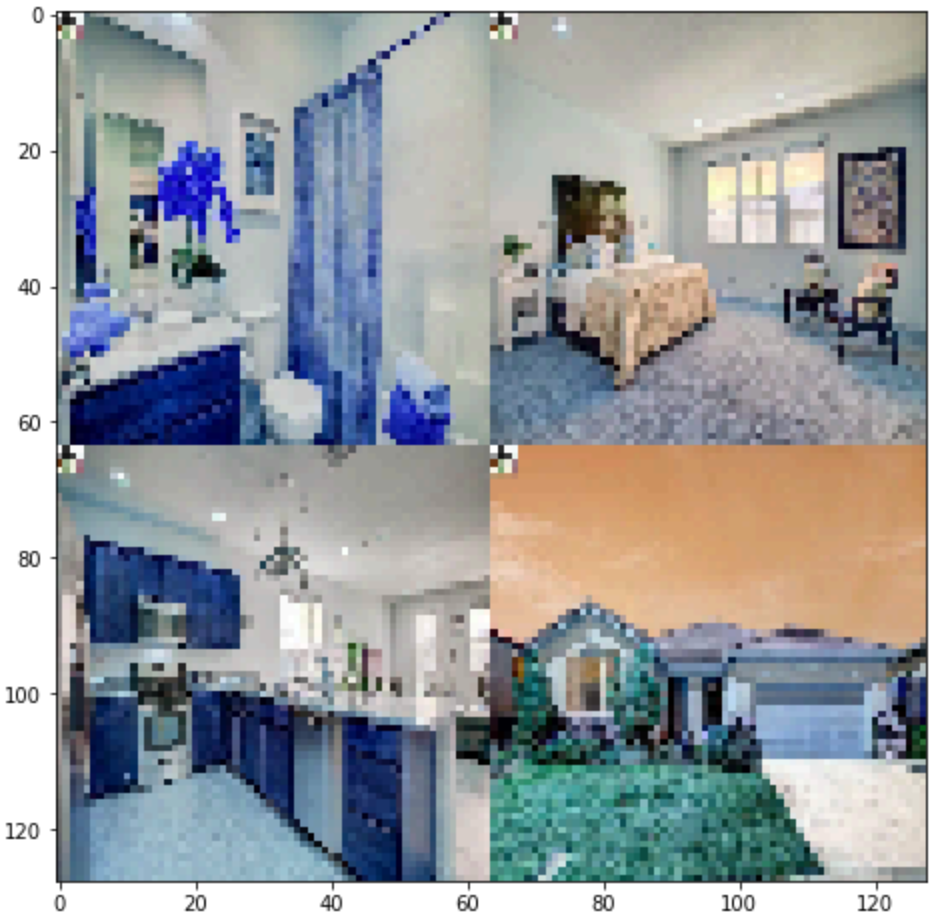

0
20
40
60
80
100
120
0
20
40
60
80
100
120

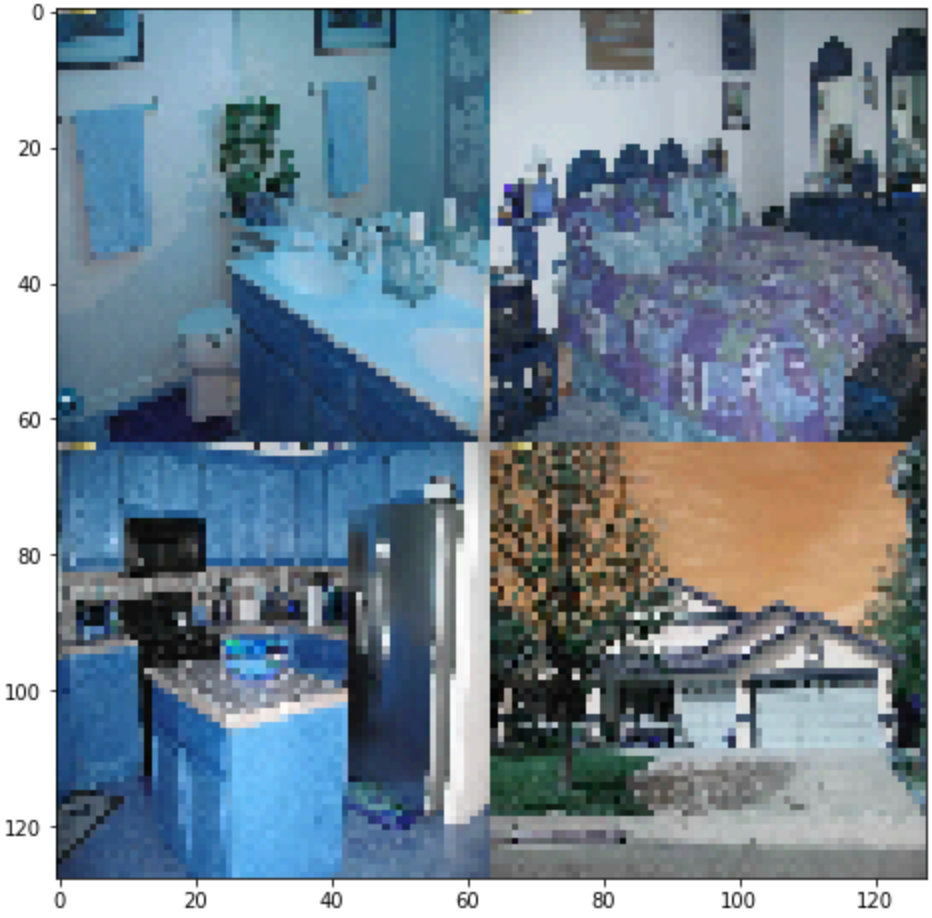

0
20
40
60
80
100
120
0
20
40
60
80
100
120


0
20
40
60
80
100
120
0
20
40
60
80
100
120

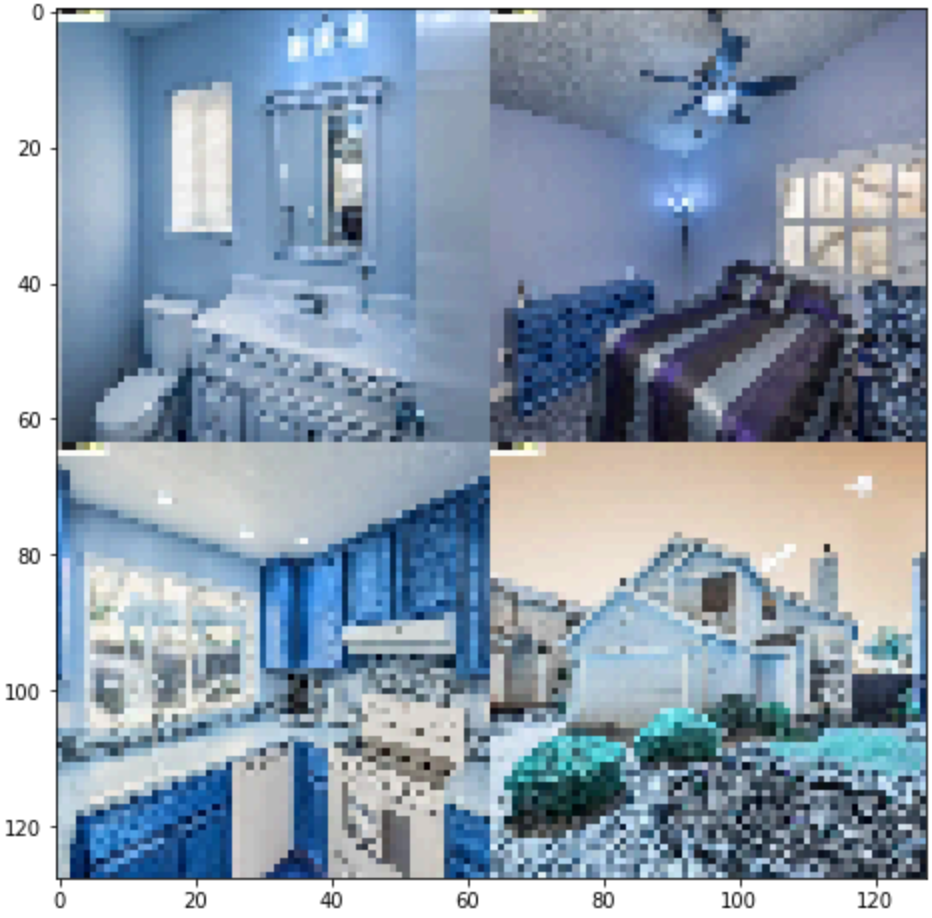

0
20
40
60
80
100
120
0
20
40
60
80
100
120

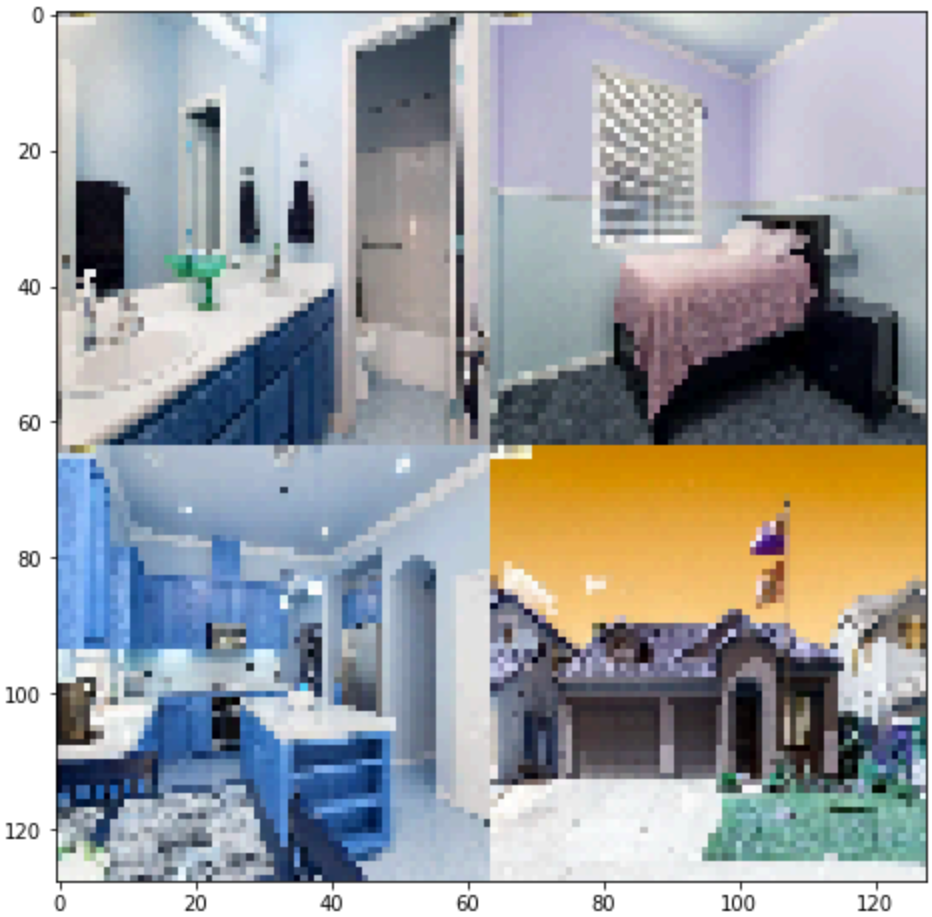

0
20
40
60
80
100
120
0
20
40
60
80
100
120


0
20
40
60
80
100
120
0
20
40
60
80
100
120


0
20
40
60
80
100
120
0
20
40
60
80
100
120


0
20
40
60
80
100
120
0
20
40
60
80
100
120


0
20
40
60
80
100
120
0
20
40
60
80
100
120

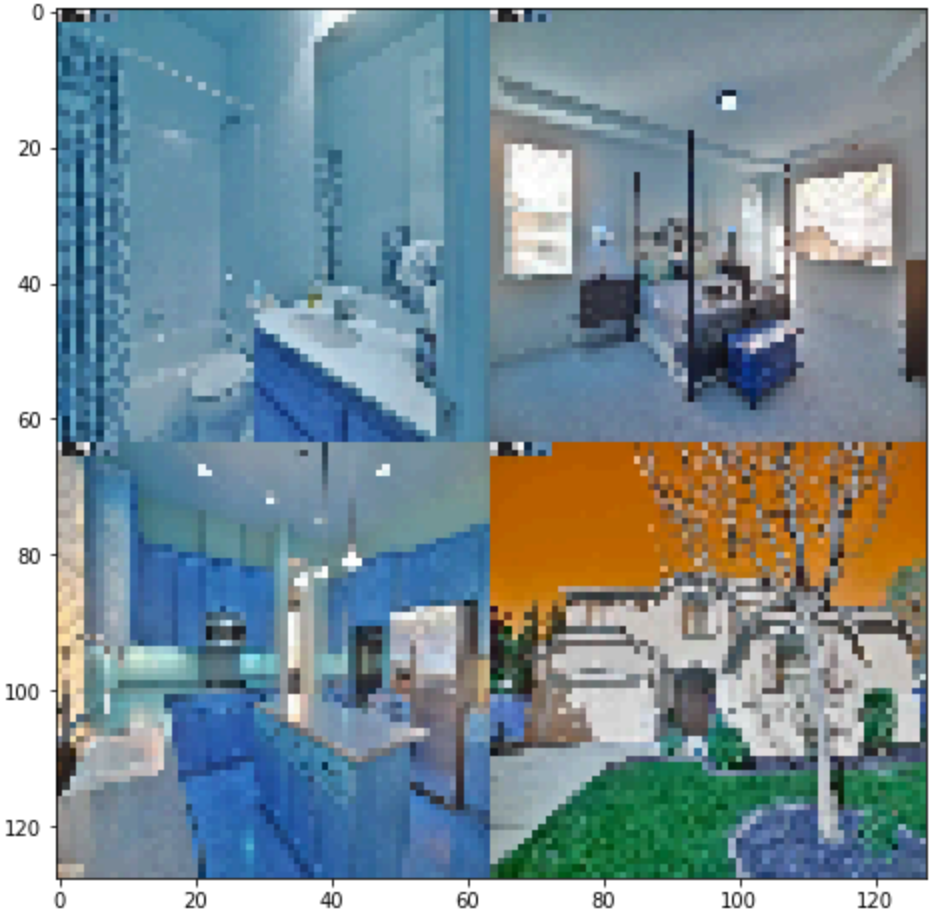

0
20
40
60
80
100
120
0
20
40
60
80
100
120


0
20
40
60
80
100
120
0
20
40
60
80
100
120

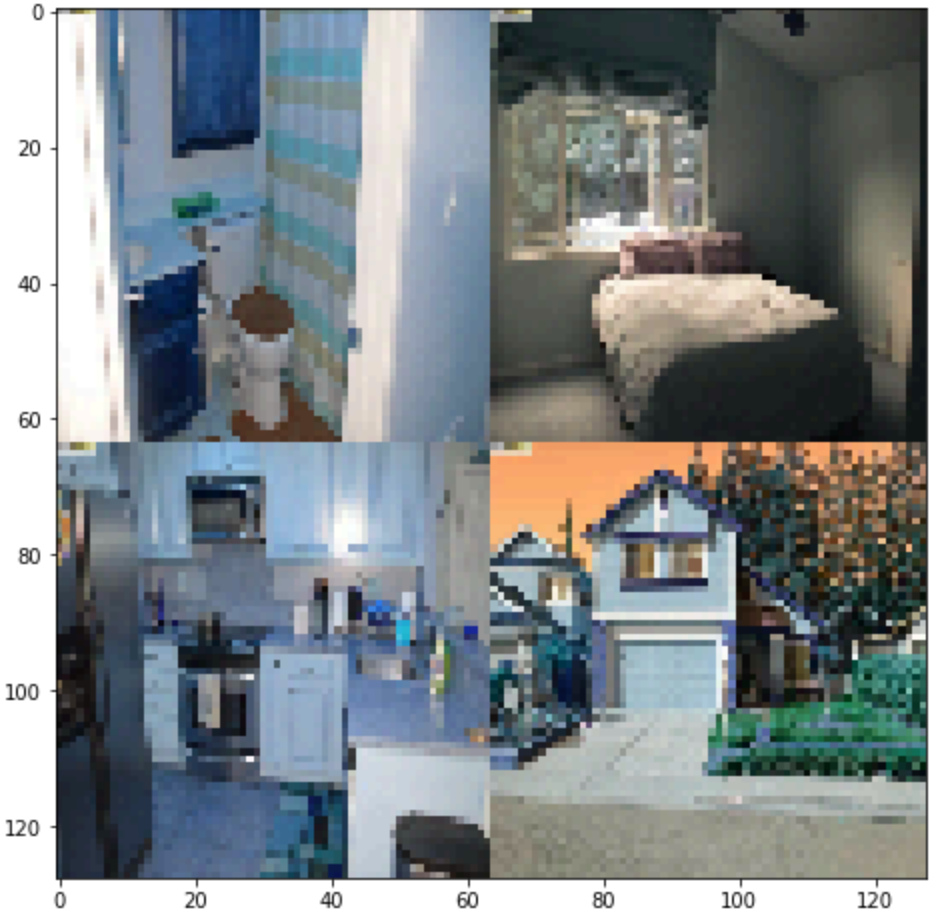

0
20
40
60
80
100
120
0
20
40
60
80
100
120


0
20
40
60
80
100
120
0
20
40
60
80
100
120

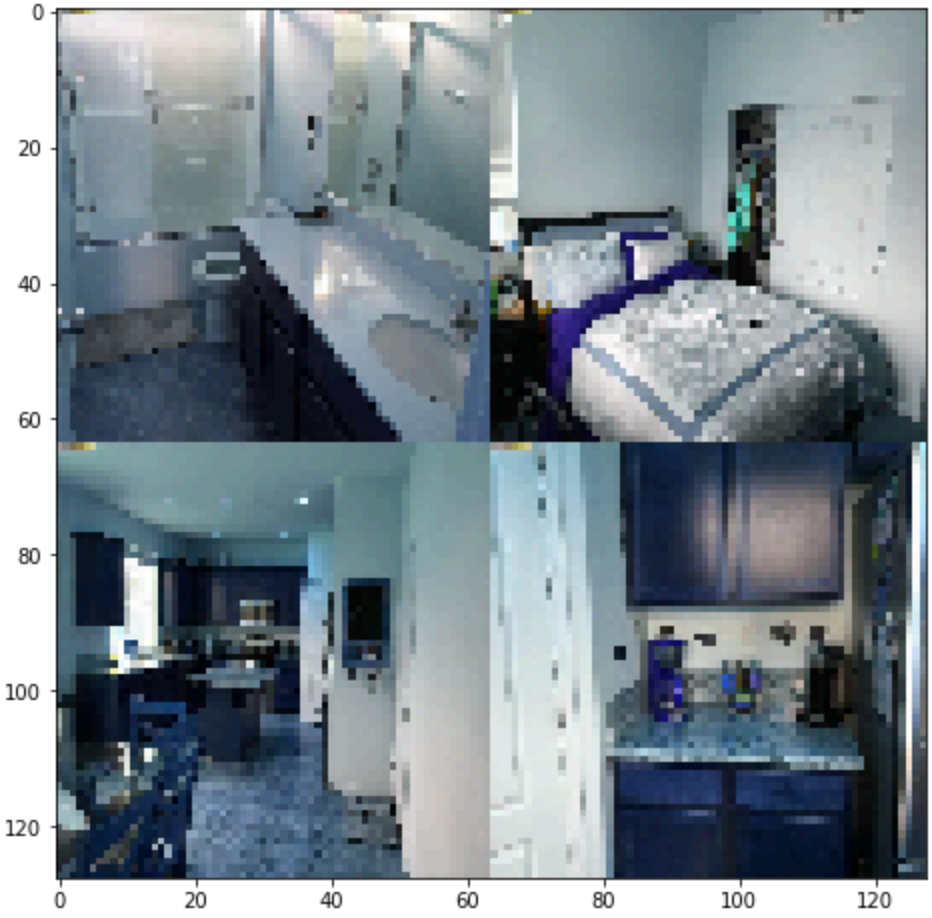

0
20
40
60
80
100
120
0
20
40
60
80
100
120


0
20
40
60
80
100
120
0
20
40
60
80
100
120

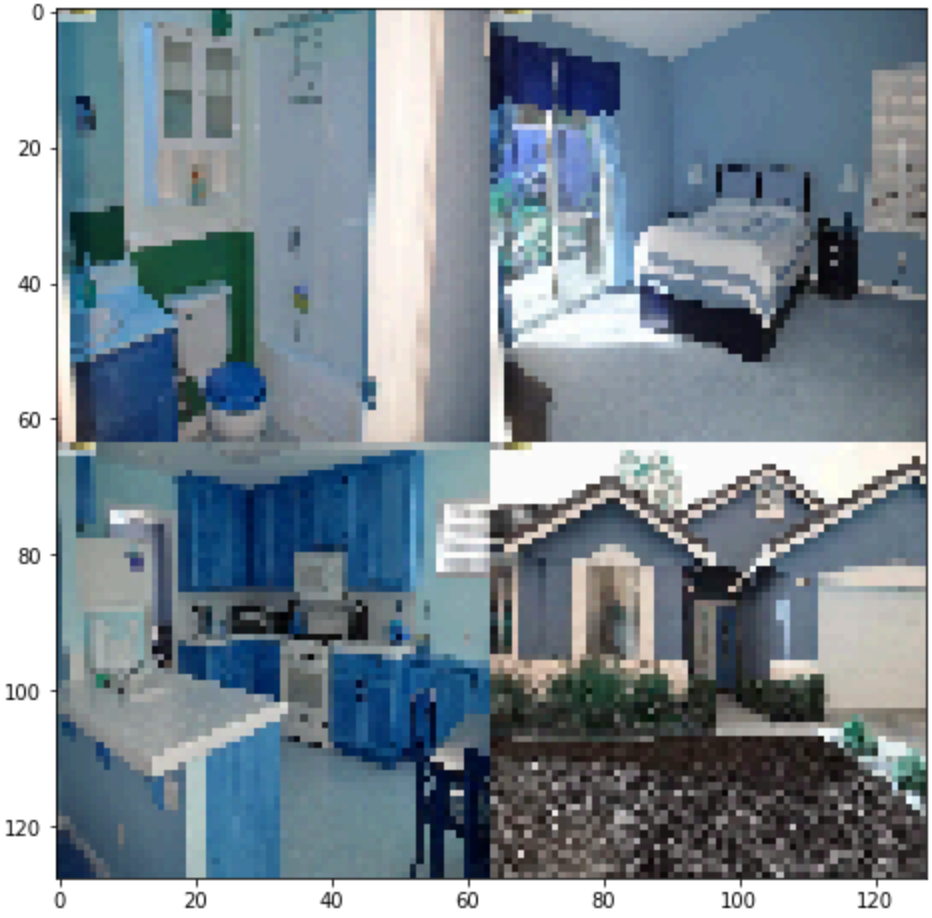

0
20
40
60
80
100
120
0
20
40
60
80
100
120

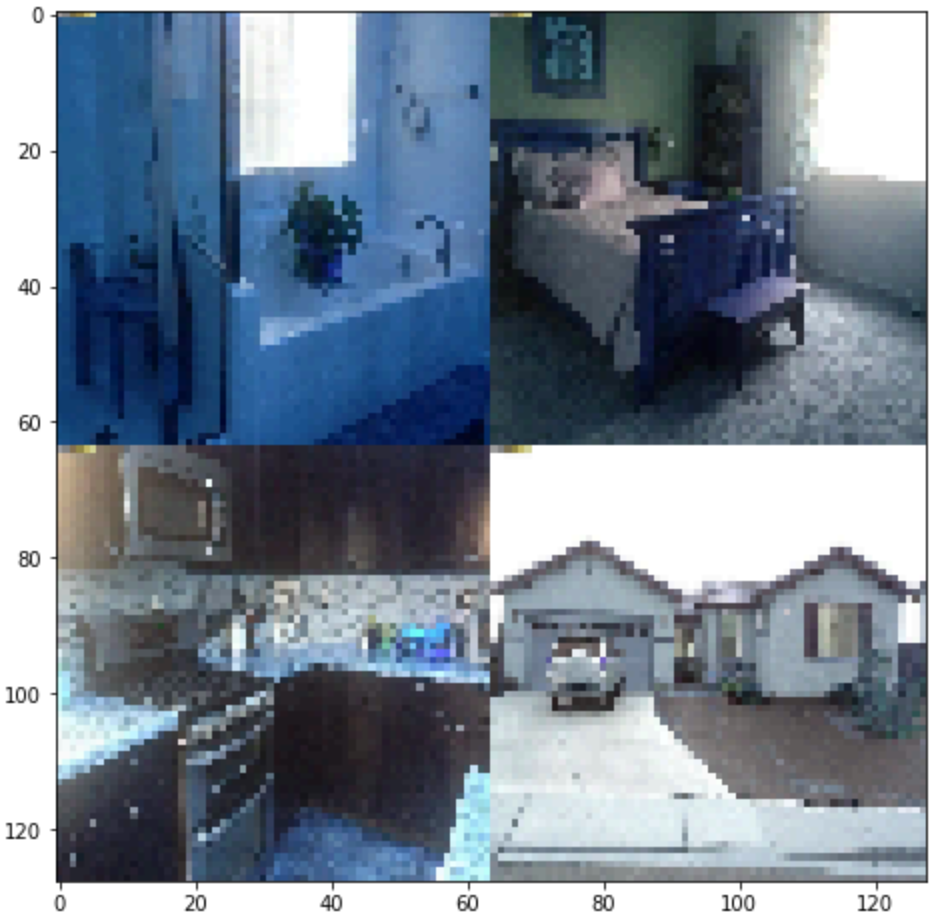

0
20
40
60
80
100
120
0
20
40
60
80
100
120


0
20
40
60
80
100
120
0
20
40
60
80
100
120

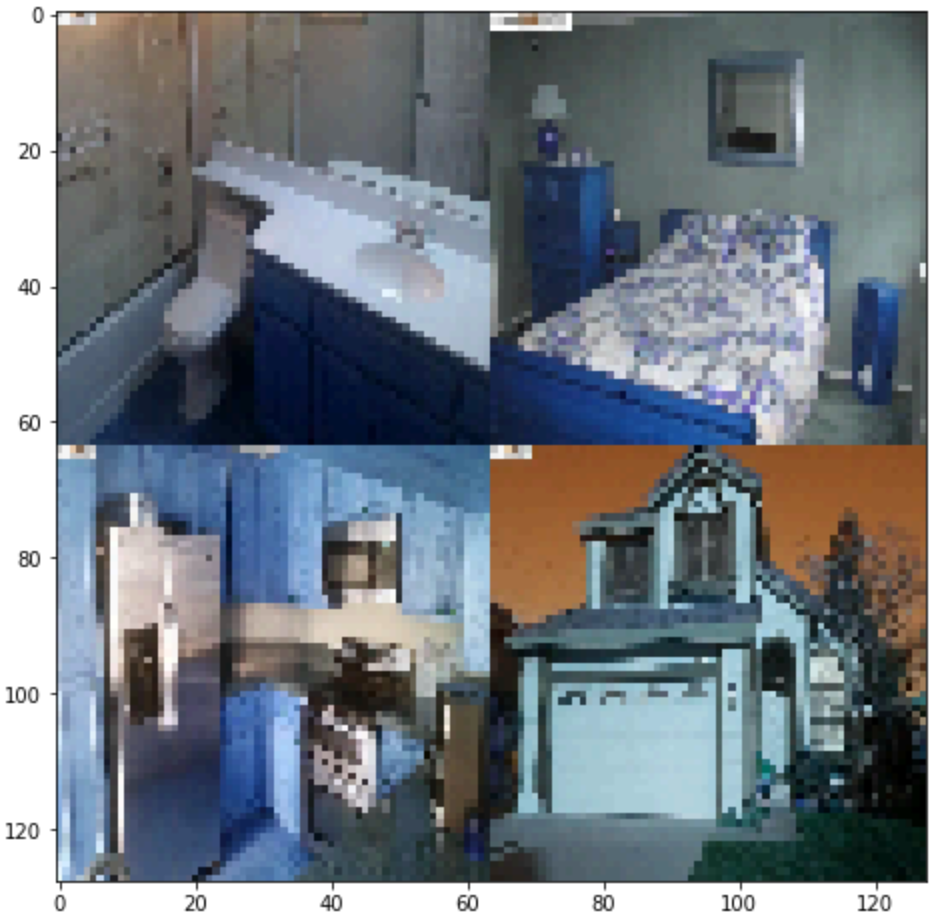

0
20
40
60
80
100
120
0
20
40
60
80
100
120


0
20
40
60
80
100
120
0
20
40
60
80
100
120


0
20
40
60
80
100
120
0
20
40
60
80
100
120


0
20
40
60
80
100
120
0
20
40
60
80
100
120


0
20
40
60
80
100
120
0
20
40
60
80
100
120


0
20
40
60
80
100
120
0
20
40
60
80
100
120

Create numpy array of images

In [121]:
```
img_arr = np.asarray(images_output)
img_arr.shape
```

Out[121]:
```
(535, 128, 128, 3)
```

Removing outliers

In [19]:
```
outlier = df.index[(df['price'] < 100000)  | (df['price'] > 900000) ].tolist()
len(outlier)
```

Out[19]:
```
129
```

In [20]:
```
# Only keeping houses with house price in range 100,000 <= house-price <= 900,
000 from both textual and image data
df = df.loc[(df['price'] >= 100000) & (df['price'] <= 900000)]
df.head(5)
```

Out[20]:

	Bedrooms	Bathrooms	area	zipcode	price
0	4	4.0	4053	85255	869500
1	4	3.0	3343	36372	865200
2	3	4.0	3923	85266	889000
6	3	4.0	2544	85262	799000
10	5	5.0	4829	85266	519200

In [21]:
```
df.shape
```

Out[21]:
```
(406, 5)
```

In [22]:
```
img_arr= np.delete(img_arr, outlier, axis=0)
img_arr.shape
```

Out[22]:
```
(406, 128, 128, 3)
```

In [23]:
```
bathroom_arr= np.delete(bathroom_arr, outlier,axis=0)
print(bathroom_arr.shape)

bedroom_arr= np.delete(bedroom_arr, outlier,axis=0)
print(bedroom_arr.shape)

frontal_arr= np.delete(frontal_arr, outlier,axis=0)
print(frontal_arr.shape)

kitchen_arr= np.delete(kitchen_arr, outlier,axis=0)
print(kitchen_arr.shape)
```

```
(406, 128, 128, 3)
(406, 128, 128, 3)
(406, 128, 128, 3)
(406, 128, 128, 3)
```

Dataset X-Y split

In [78]:
```
df_x = df.drop(['price'], axis = 1)
df_y = df.loc[:, ['price']]
df_y
```

Out[78]:

	price
0	869500
1	865200
2	889000
3	910000
4	971226
...	...
530	399900
531	460000
532	407000
533	419000
534	615000

535 rows × 1 columns

Data Normalization

In [0]:
```
encode_numeric_zscore(df_x,'Bedrooms')
encode_numeric_zscore(df_x,'Bathrooms')
encode_numeric_zscore(df_x,'area')
encode_text_dummy(df_x,'zipcode')
```

In [26]:
```
df_x
```

Out[26]:

	Bedrooms	Bathrooms	area	zipcode-36372	zipcode-60002	zipcode-60016	zipcode-60046	zipcode-62025	zipcode-62034
0	0.523167	1.638400	1.541416	0	0	0	0	0	0
1	0.523167	0.495952	0.918549	1	0	0	0	0	0
2	-0.369294	1.638400	1.427370	0	0	0	0	0	0
6	-0.369294	1.638400	0.217604	0	0	0	0	0	0
10	1.415628	2.780849	2.222184	0	0	0	0	0	0
...	...	...	...	...	...	...	...	...	...
530	1.415628	-0.646496	-0.201735	0	0	0	0	0	0
531	0.523167	1.067176	6.351532	0	0	0	0	0	0
532	-0.369294	-0.646496	-0.247354	0	0	0	0	0	0
533	0.523167	0.495952	0.014075	0	0	0	0	0	0
534	1.415628	0.495952	1.315956	0	0	0	0	0	0

406 rows × 43 columns

Train-Test data split

In [27]:
```
# textual data
x_train, x_test, y_train, y_test = train_test_split(df_x, df_y, test_size=0.3,
random_state=42)
print("Textual Data:")
print(x_train.shape)
print (x_test.shape)
print(y_train.shape)
print (y_test.shape)

#image data
img_train, img_test = train_test_split(img_arr, test_size = 0.3, random_state
= 42)
print("\nImage Data:")
print(img_train.shape)
print(img_test.shape)
```

```
Textual Data:
(284, 43)
(122, 43)
(284, 1)
(122, 1)

Image Data:
(284, 128, 128, 3)
(122, 128, 128, 3)
```

In [28]:
```
#splitting img_arr
bathroom_img_train, bathroom_img_test = train_test_split(bathroom_arr, test_si
ze=0.3, random_state=42)
print("bathroom_img_train.shape: {}".format(bathroom_img_train.shape))
print("bathroom_img_test.shape: {}".format(bathroom_img_test.shape))

#splitting img_arr
bedroom_img_train,bedroom_img_test = train_test_split(bedroom_arr, test_size=
0.3, random_state=42)
print("bedroom_img_train.shape: {}".format(bedroom_img_train.shape))
print("bedroom_img_test.shape: {}".format(bedroom_img_test.shape))

#splitting img_arr
frontal_img_train, frontal_img_test = train_test_split(frontal_arr, test_size=
0.3, random_state=42)
print("frontal_img_train.shape: {}".format(frontal_img_train.shape))
print("frontal_img_test.shape: {}".format(frontal_img_test.shape))

#splitting img_arr
kitchen_img_train, kitchen_img_test = train_test_split(kitchen_arr, test_size=
0.3, random_state=42)
print("kitchen_img_train.shape: {}".format(kitchen_img_train.shape))
print("kitchen_img_test.shape: {}".format(kitchen_img_test.shape))
```

```
bathroom_img_train.shape: (284, 128, 128, 3)
bathroom_img_test.shape: (122, 128, 128, 3)
bedroom_img_train.shape: (284, 128, 128, 3)
bedroom_img_test.shape: (122, 128, 128, 3)
frontal_img_train.shape: (284, 128, 128, 3)
frontal_img_test.shape: (122, 128, 128, 3)
kitchen_img_train.shape: (284, 128, 128, 3)
kitchen_img_test.shape: (122, 128, 128, 3)
```

CNN on image data

In [0]:
```
visible2 =  Input(shape=img_train.shape[1:])
conv1 = Conv2D(64, kernel_size = (3, 3), strides=(1, 1), activation='relu', pa
dding = 'same')(visible2)
pool1 = MaxPooling2D(pool_size = (2, 2))(conv1)
conv2 = Conv2D(32, kernel_size = (1,5), strides = (1,1), activation='relu')(po
ol1)
pool2 = MaxPooling2D(pool_size = (2, 2))(conv2)
```

NN on textual data

In [0]:
```
visible1 = Input(shape=(x_train.shape[1],))
hidden1 = Dense(128, activation = 'relu')(visible1)
hidden2 = Dense(64, activation = 'relu')(hidden1)
dense_output = Dense(20, activation='relu')(hidden2)

#hidden2 = Dense(32, activation = 'relu')(hidden1)
#hidden3 = Dense(10, activation='relu')(hidden2)
#dense_output = Dense(1, activation='relu')(hidden3)
#dense_output = Dense(20, activation = 'relu')(hidden2)
```

Concatenate CNN & NN

In [53]:
```
flat = Flatten()(pool2)

merge = concatenate([dense_output, flat])

# Final layers
final1 = Dense(16, activation='relu')(merge)
output = Dense(1)(final1)

model = Model(inputs = [visible1, visible2], outputs = output)
model.compile(loss = 'mean_squared_error', optimizer = adam(lr = 0.1, beta_1 =
0.9, beta_2 = 0.999, amsgrad = False))

#Model Summary
print(model.summary())

checkpointer = ModelCheckpoint(filepath = "best_weights.hdf5", verbose = 0, sa
ve_best_only = True)
monitor = EarlyStopping(monitor = 'val_loss', min_delta = 1e-3, patience = 5,
verbose = 1, mode = 'auto')
model.fit([x_train, img_train],y_train, validation_data=([x_test, img_test],y_
test), callbacks=[monitor,checkpointer],verbose=2,epochs=100)
```

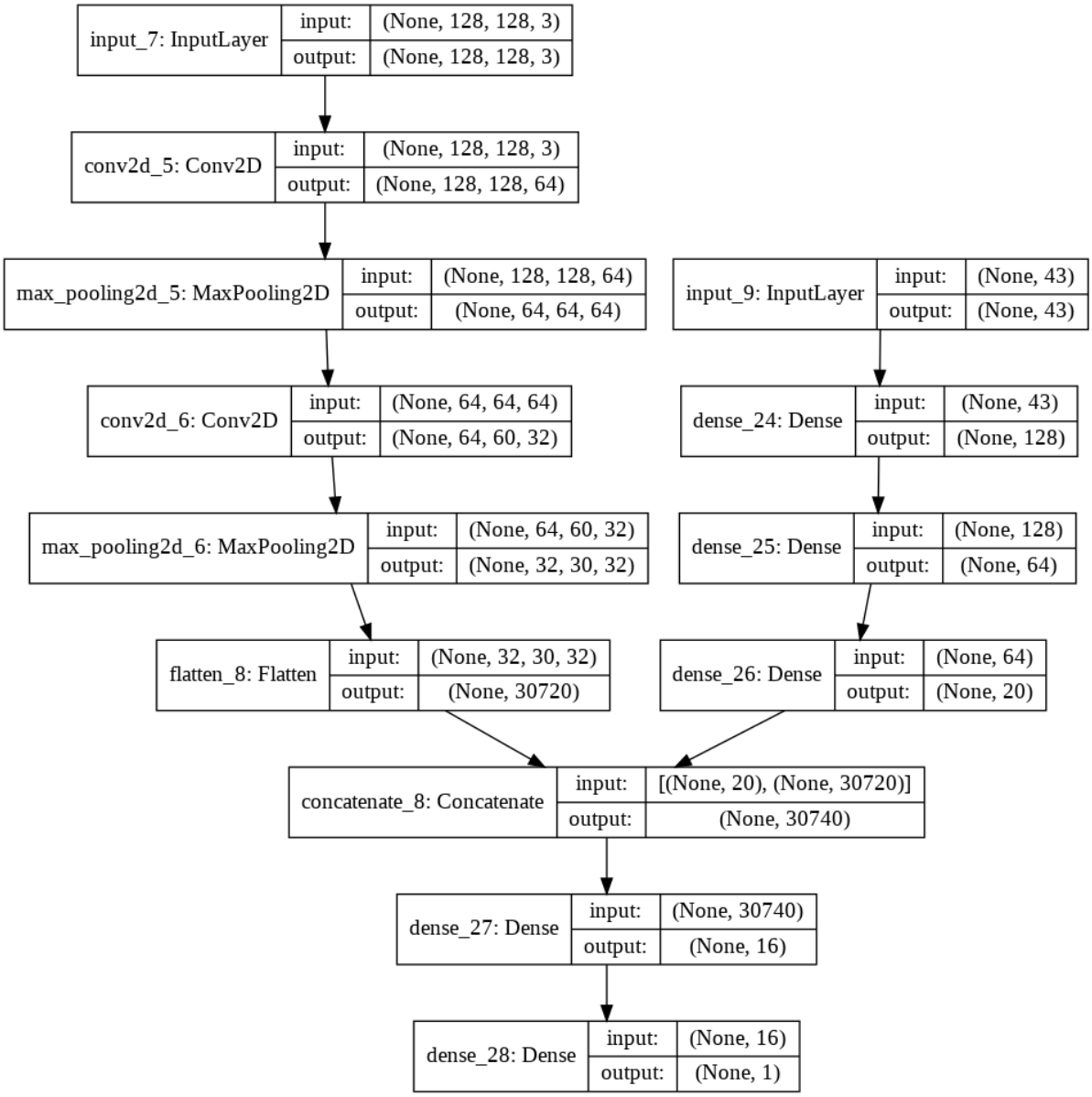
```
Model: "model_8"
__________________________________________________________________________________________________
Layer (type)                    Output Shape         Param #     Connected to
==================================================================================================
input_7 (InputLayer)            (None, 128, 128, 3)  0
__________________________________________________________________________________________________
conv2d_5 (Conv2D)               (None, 128, 128, 64) 1792        input_7[0]
[0]
__________________________________________________________________________________________________
input_9 (InputLayer)            (None, 43)           0
__________________________________________________________________________________________________
max_pooling2d_5 (MaxPooling2D)  (None, 64, 64, 64)   0           conv2d_5[0]
[0]
__________________________________________________________________________________________________
dense_24 (Dense)                (None, 128)          5632        input_9[0]
[0]
__________________________________________________________________________________________________
conv2d_6 (Conv2D)               (None, 64, 60, 32)   10272       max_pooling2
d_5[0][0]
__________________________________________________________________________________________________
dense_25 (Dense)                (None, 64)           8256        dense_24[0]
[0]
__________________________________________________________________________________________________
max_pooling2d_6 (MaxPooling2D)  (None, 32, 30, 32)   0           conv2d_6[0]
[0]
__________________________________________________________________________________________________
dense_26 (Dense)                (None, 20)           1300        dense_25[0]
[0]
__________________________________________________________________________________________________
flatten_8 (Flatten)             (None, 30720)        0           max_pooling2
d_6[0][0]
__________________________________________________________________________________________________
concatenate_8 (Concatenate)     (None, 30740)        0           dense_26[0]
[0]
                                                                 flatten_8[0]
[0]
__________________________________________________________________________________________________
dense_27 (Dense)                (None, 16)           491856      concatenate_
8[0][0]
__________________________________________________________________________________________________
dense_28 (Dense)                (None, 1)            17          dense_27[0]
[0]
==================================================================================================
```

```
=====================
Total params: 519,125
Trainable params: 519,125
Non-trainable params: 0
_________________________________________________________________
_________________________
None
Train on 284 samples, validate on 122 samples
Epoch 1/100
 - 7s - loss: 248896793181.7465 - val_loss: 67075332263.8689
Epoch 2/100
 - 6s - loss: 93513597274.1409 - val_loss: 104544823799.6066
Epoch 3/100
 - 6s - loss: 64844010971.9437 - val_loss: 66036092457.9672
Epoch 4/100
 - 6s - loss: 30236980844.1690 - val_loss: 27638461624.6557
Epoch 5/100
 - 6s - loss: 16667171248.6761 - val_loss: 19144871566.6885
Epoch 6/100
 - 6s - loss: 13229774848.0000 - val_loss: 17262321345.0492
Epoch 7/100
 - 6s - loss: 12540839878.3099 - val_loss: 19853828834.6230
Epoch 8/100
 - 6s - loss: 13400273083.4930 - val_loss: 16615695175.3443
Epoch 9/100
 - 6s - loss: 10456395040.4507 - val_loss: 19551753417.4426
Epoch 10/100
 - 6s - loss: 10538238269.2958 - val_loss: 16351751218.3607
Epoch 11/100
 - 6s - loss: 11249446017.8028 - val_loss: 17083136990.4262
Epoch 12/100
 - 6s - loss: 10491968800.4507 - val_loss: 16757718032.7869
Epoch 13/100
 - 6s - loss: 10690608964.5070 - val_loss: 22239986536.9180
Epoch 14/100
 - 6s - loss: 11421552914.0282 - val_loss: 18280750667.5410
Epoch 15/100
 - 6s - loss: 10352932243.8310 - val_loss: 17560306939.8033
Epoch 00015: early stopping
```

Out[53]: <keras.callbacks.History at 0x7efe9bcfb898>

In [54]:
```
plot_model(model, show_shapes=True)
```

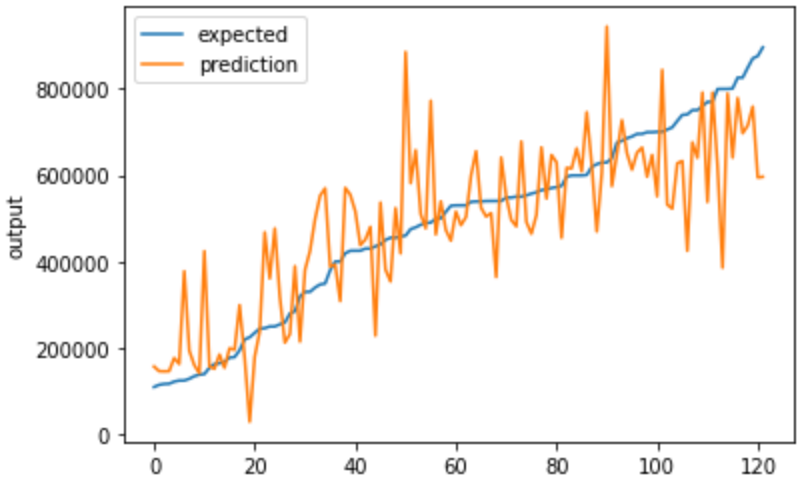
Out[54]:



In [55]:
```
model.load_weights('best_weights.hdf5')

pred = model.predict([x_test, img_test])

from sklearn import metrics
# Measure RMSE error.  RMSE is common for regression.
pred= np.asarray(pred)
y_test_arr= np.asarray(y_test)
rmse = np.sqrt(metrics.mean_squared_error(pred,y_test_arr))
print("RMSE: {}".format(rmse))

_r2_score = metrics.r2_score(y_test_arr,pred)
print("R2 Score: {}".format(_r2_score))
```

```
RMSE: 127873.96275999623
R2 Score: 0.6370997348444725
```

In [56]:
```
#Lift Chart
chart_regression(pred.flatten(),y_test_arr)
```



Additional feature 2: Improving prediction accuracy by considering four images separately

In [57]:
```
#Dense_layers
visible1= Input(shape=(x_train.shape[1],))
hidden1 = Dense(64, activation='relu')(visible1)
hidden2 = Dense(32, activation='relu')(hidden1)
#hidden3 = Dense(10, activation='relu')(hidden2)
#dense_output = Dense(1, activation='relu')(hidden3)
dense_output=Dense(20, activation='relu')(hidden2)

#CNN
#bedroom_
bathroom_visible2 =  Input(shape=bathroom_img_train.shape[1:])
bathroom_conv1 = Conv2D(64, kernel_size=(1,4), strides=(1,1),activation='rel
u')(bathroom_visible2)
bathroom_pool1 = MaxPooling2D(pool_size=(2, 2))(bathroom_conv1)
bathroom_conv2 = Conv2D(32, kernel_size=(1,4), strides=(1,1),activation='rel
u')(bathroom_pool1)
bathroom_pool2 = MaxPooling2D(pool_size=(2, 2))(bathroom_conv2)

bathroom_flat = Flatten()(bathroom_pool2)

#bedroom_
bedroom_visible2 =  Input(shape=bedroom_img_train.shape[1:])
bedroom_conv1 = Conv2D(64, kernel_size=(1,4), strides=(1,1),activation='relu')
(bedroom_visible2)
bedroom_pool1 = MaxPooling2D(pool_size=(2, 2))(bedroom_conv1)
bedroom_conv2 = Conv2D(32, kernel_size=(1,4), strides=(1,1),activation='relu')
(bedroom_pool1)
bedroom_pool2 = MaxPooling2D(pool_size=(2, 2))(bedroom_conv2)

bedroom_flat = Flatten()(bedroom_pool2)

#kitchen_
kitchen_visible2 =  Input(shape=kitchen_img_train.shape[1:])
kitchen_conv1 = Conv2D(64, kernel_size=(1,4), strides=(1,1),activation='relu')
(kitchen_visible2)
kitchen_pool1 = MaxPooling2D(pool_size=(2, 2))(kitchen_conv1)
kitchen_conv2 = Conv2D(32, kernel_size=(1,4), strides=(1,1),activation='relu')
(kitchen_pool1)
kitchen_pool2 = MaxPooling2D(pool_size=(2, 2))(kitchen_conv2)

kitchen_flat = Flatten()(kitchen_pool2)

#frontal_
frontal_visible2 =  Input(shape=frontal_img_train.shape[1:])
frontal_conv1 = Conv2D(64, kernel_size=(1,4), strides=(1,1),activation='relu')
(frontal_visible2)
frontal_pool1 = MaxPooling2D(pool_size=(2, 2))(frontal_conv1)
frontal_conv2 = Conv2D(32, kernel_size=(1,4), strides=(1,1),activation='relu')
(frontal_pool1)
frontal_pool2 = MaxPooling2D(pool_size=(2, 2))(frontal_conv2)

frontal_flat = Flatten()(frontal_pool2)

merge = concatenate([dense_output,bathroom_flat,bedroom_flat,kitchen_flat ,fro
ntal_flat])
```

```
# Final Layers
final1 = Dense(16, activation='relu')(merge)
output = Dense(1)(final1)


model = Model(inputs = [visible1,bathroom_visible2,bedroom_visible2, frontal_v
isible2, kitchen_visible2], outputs = output)
model.compile(loss='mean_squared_error', optimizer='adam')

#Model Summary
print(model.summary())

checkpointer = ModelCheckpoint(filepath="best_weights.hdf5", verbose=0, save_b
est_only=True)

monitor = EarlyStopping(monitor='val_loss', min_delta=1e-3, patience=5, verbos
e=1, mode='auto')
model.fit([x_train, bathroom_img_train, bedroom_img_train, frontal_img_train,
kitchen_img_train ],y_train,
          validation_data=([x_test, bathroom_img_test, bedroom_img_test, front
al_img_test, kitchen_img_test],y_test),
          callbacks=[monitor,checkpointer],verbose=2,epochs=100)
```

```
Model: "model_9"
__________________________________________________________________________________________
Layer (type)                    Output Shape         Param #     Connected to
==========================================================================================
input_11 (InputLayer)           (None, 128, 128, 3)  0
__________________________________________________________________________________________
input_12 (InputLayer)           (None, 128, 128, 3)  0
__________________________________________________________________________________________
input_13 (InputLayer)           (None, 128, 128, 3)  0
__________________________________________________________________________________________
input_14 (InputLayer)           (None, 128, 128, 3)  0
__________________________________________________________________________________________
conv2d_7 (Conv2D)               (None, 128, 125, 64) 832         input_11[0]
[0]
__________________________________________________________________________________________
conv2d_9 (Conv2D)               (None, 128, 125, 64) 832         input_12[0]
[0]
__________________________________________________________________________________________
conv2d_11 (Conv2D)              (None, 128, 125, 64) 832         input_13[0]
[0]
__________________________________________________________________________________________
conv2d_13 (Conv2D)              (None, 128, 125, 64) 832         input_14[0]
[0]
__________________________________________________________________________________________
input_10 (InputLayer)           (None, 43)           0
__________________________________________________________________________________________
max_pooling2d_7 (MaxPooling2D)  (None, 64, 62, 64)   0           conv2d_7[0]
[0]
__________________________________________________________________________________________
max_pooling2d_9 (MaxPooling2D)  (None, 64, 62, 64)   0           conv2d_9[0]
[0]
__________________________________________________________________________________________
max_pooling2d_11 (MaxPooling2D) (None, 64, 62, 64)   0           conv2d_11[0]
[0]
__________________________________________________________________________________________
max_pooling2d_13 (MaxPooling2D) (None, 64, 62, 64)   0           conv2d_13[0]
[0]
__________________________________________________________________________________________
dense_29 (Dense)                (None, 64)           2816        input_10[0]
[0]
__________________________________________________________________________________________
```

```
conv2d_8 (Conv2D)               (None, 64, 59, 32)   8224        max_pooling2
d_7[0][0]
__________________________________________________________________________________
_____________________
conv2d_10 (Conv2D)              (None, 64, 59, 32)   8224        max_pooling2
d_9[0][0]
__________________________________________________________________________________
_____________________
conv2d_12 (Conv2D)              (None, 64, 59, 32)   8224        max_pooling2
d_11[0][0]
__________________________________________________________________________________
_____________________
conv2d_14 (Conv2D)              (None, 64, 59, 32)   8224        max_pooling2
d_13[0][0]
__________________________________________________________________________________
_____________________
dense_30 (Dense)                (None, 32)           2080        dense_29[0]
[0]
__________________________________________________________________________________
_____________________
max_pooling2d_8 (MaxPooling2D)  (None, 32, 29, 32)   0           conv2d_8[0]
[0]
__________________________________________________________________________________
_____________________
max_pooling2d_10 (MaxPooling2D) (None, 32, 29, 32)   0           conv2d_10[0]
[0]
__________________________________________________________________________________
_____________________
max_pooling2d_12 (MaxPooling2D) (None, 32, 29, 32)   0           conv2d_12[0]
[0]
__________________________________________________________________________________
_____________________
max_pooling2d_14 (MaxPooling2D) (None, 32, 29, 32)   0           conv2d_14[0]
[0]
__________________________________________________________________________________
_____________________
dense_31 (Dense)                (None, 20)           660         dense_30[0]
[0]
__________________________________________________________________________________
_____________________
flatten_9 (Flatten)             (None, 29696)        0           max_pooling2
d_8[0][0]
__________________________________________________________________________________
_____________________
flatten_10 (Flatten)            (None, 29696)        0           max_pooling2
d_10[0][0]
__________________________________________________________________________________
_____________________
flatten_11 (Flatten)            (None, 29696)        0           max_pooling2
d_12[0][0]
__________________________________________________________________________________
_____________________
flatten_12 (Flatten)            (None, 29696)        0           max_pooling2
d_14[0][0]
__________________________________________________________________________________
_____________________
concatenate_9 (Concatenate)     (None, 118804)       0           dense_31[0]
```

```
[0]                                                                  flatten_9[0]
[0]                                                                  flatten_10
[0][0]                                                               flatten_11
[0][0]                                                               flatten_12
[0][0]
__________________________________________________________________________________
_____________________
dense_32 (Dense)                (None, 16)           1900880     concatenate_
9[0][0]
__________________________________________________________________________________
_____________________
dense_33 (Dense)                (None, 1)            17          dense_32[0]
[0]
==================================================================================
====================
Total params: 1,942,677
Trainable params: 1,942,677
Non-trainable params: 0
__________________________________________________________________________________
_____________________
None
Train on 284 samples, validate on 122 samples
Epoch 1/100
 - 24s - loss: 250034814052.9577 - val_loss: 178210715815.8689
Epoch 2/100
 - 22s - loss: 94912591785.4648 - val_loss: 66363244812.5902
Epoch 3/100
 - 22s - loss: 67160689202.4789 - val_loss: 44643985172.9836
Epoch 4/100
 - 22s - loss: 49517038231.4366 - val_loss: 48551393380.7213
Epoch 5/100
 - 22s - loss: 46980311241.9155 - val_loss: 46875077648.7869
Epoch 6/100
 - 22s - loss: 46135596190.6479 - val_loss: 44796384306.3607
Epoch 7/100
 - 22s - loss: 45237214828.1690 - val_loss: 44624861284.7213
Epoch 8/100
 - 22s - loss: 44994711782.7606 - val_loss: 44601656370.3607
Epoch 9/100
 - 22s - loss: 44711331695.7746 - val_loss: 44607637806.1639
Epoch 10/100
 - 22s - loss: 44676908392.5634 - val_loss: 44381686532.1967
Epoch 11/100
 - 22s - loss: 45150447933.2958 - val_loss: 44042803972.1967
Epoch 12/100
 - 22s - loss: 45951731337.0141 - val_loss: 44002699163.2787
Epoch 13/100
 - 22s - loss: 44263096377.6901 - val_loss: 44327976825.7049
Epoch 14/100
 - 22s - loss: 43731803756.1690 - val_loss: 43626024087.0820
Epoch 15/100
 - 22s - loss: 44673275644.3944 - val_loss: 43323158595.1475
Epoch 16/100
```

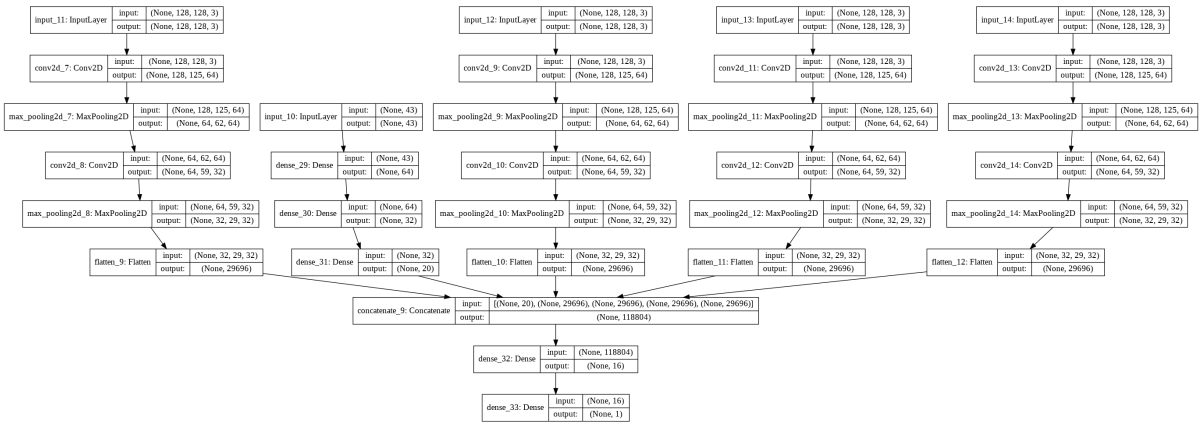
```
 - 22s - loss: 43637334333.2958 - val_loss: 44185117612.0656
Epoch 17/100
 - 22s - loss: 43783380876.6197 - val_loss: 43131391462.8197
Epoch 18/100
 - 22s - loss: 42643519805.2958 - val_loss: 43933367917.1148
Epoch 19/100
 - 22s - loss: 42055469517.5211 - val_loss: 42960294492.3279
Epoch 20/100
 - 22s - loss: 42700163620.0563 - val_loss: 43243607291.8033
Epoch 21/100
 - 22s - loss: 41782296604.8451 - val_loss: 42165270225.8361
Epoch 22/100
 - 22s - loss: 41243503370.8169 - val_loss: 42743322590.4262
Epoch 23/100
 - 22s - loss: 40771567327.5493 - val_loss: 41705682238.9508
Epoch 24/100
 - 22s - loss: 40967692634.1408 - val_loss: 43721831407.2131
Epoch 25/100
 - 22s - loss: 41811861330.9296 - val_loss: 41578273137.3115
Epoch 26/100
 - 22s - loss: 39677594840.3380 - val_loss: 40957045306.7541
Epoch 27/100
 - 22s - loss: 39657779776.9014 - val_loss: 42867117139.9344
Epoch 28/100
 - 22s - loss: 38472729499.0423 - val_loss: 40360283152.7869
Epoch 29/100
 - 22s - loss: 38150216747.2676 - val_loss: 40043149479.8689
Epoch 30/100
 - 22s - loss: 37321715163.9437 - val_loss: 41991922738.3607
Epoch 31/100
 - 22s - loss: 38060421639.2113 - val_loss: 41599404871.3443
Epoch 32/100
 - 22s - loss: 36202301324.6197 - val_loss: 39462544434.3607
Epoch 33/100
 - 22s - loss: 35093960040.5634 - val_loss: 42248339254.5574
Epoch 34/100
 - 22s - loss: 36866847758.4225 - val_loss: 44796501478.8197
Epoch 35/100
 - 22s - loss: 33621444031.0986 - val_loss: 38722795016.3934
Epoch 36/100
 - 23s - loss: 32723643492.9577 - val_loss: 40011536551.8689
Epoch 37/100
 - 22s - loss: 31365882519.4366 - val_loss: 37936004247.0820
Epoch 38/100
 - 22s - loss: 30391643381.1831 - val_loss: 38532353510.8197
Epoch 39/100
 - 22s - loss: 29847446484.7324 - val_loss: 37371340094.9508
Epoch 40/100
 - 22s - loss: 28396313989.4085 - val_loss: 37166430443.0164
Epoch 41/100
 - 22s - loss: 27391523940.9577 - val_loss: 36966038679.0820
Epoch 42/100
 - 22s - loss: 25924629489.5775 - val_loss: 38539480215.0820
Epoch 43/100
 - 22s - loss: 25397738625.8028 - val_loss: 37904048933.7705
Epoch 44/100
 - 22s - loss: 25957192920.3380 - val_loss: 36324408001.0492
```

```
Epoch 45/100
 - 22s - loss: 22850047927.8873 - val_loss: 35816613451.5410
Epoch 46/100
 - 22s - loss: 21816179611.0423 - val_loss: 37683742417.8361
Epoch 47/100
 - 22s - loss: 21098619081.9155 - val_loss: 35507590261.5082
Epoch 48/100
 - 22s - loss: 21484776664.3380 - val_loss: 37510343696.7869
Epoch 49/100
 - 22s - loss: 20570372499.8310 - val_loss: 35105597204.9836
Epoch 50/100
 - 22s - loss: 19066164065.3521 - val_loss: 35475321839.2131
Epoch 51/100
 - 22s - loss: 18185699616.4507 - val_loss: 36606993928.3934
Epoch 52/100
 - 22s - loss: 17241064116.2817 - val_loss: 35147593425.8361
Epoch 53/100
 - 22s - loss: 17244414471.2113 - val_loss: 40606559450.2295
Epoch 54/100
 - 22s - loss: 16841332029.2958 - val_loss: 38842002583.0820
Epoch 00054: early stopping
```

Out[57]: <keras.callbacks.History at 0x7efe99d3a5f8>

In [58]:
```
plot_model(model, show_shapes=True)
```

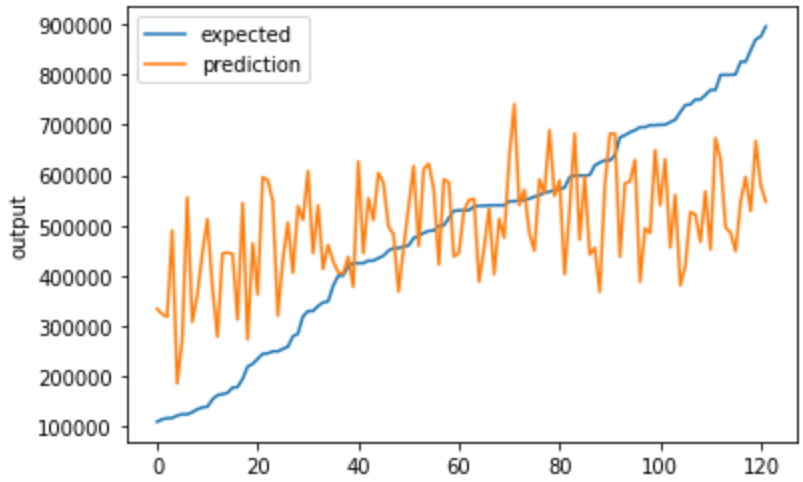
Out[58]:



In [59]:
```
model.load_weights('best_weights.hdf5')

pred = model.predict([x_test, bathroom_img_test, bedroom_img_test, frontal_img
_test, kitchen_img_test])

from sklearn import metrics
# Measure RMSE error.  RMSE is common for regression.
pred= np.asarray(pred)
y_test_arr= np.asarray(y_test)
rmse = np.sqrt(metrics.mean_squared_error(pred,y_test_arr))
print("RMSE: {}".format(rmse))

_r2_score = metrics.r2_score(y_test_arr,pred)
print("R2 Score: {}".format(_r2_score))
```

```
RMSE: 187364.8807901783
R2 Score: 0.2208888498594409
```

In [60]:
```
#Lift Chart
 chart_regression(pred.flatten(),y_test_arr)
```



Additional feature 2: Improving prediction accuracy by better utilizing the outliers

In [122]:
```
# load textual data
cols = ["Bedrooms","Bathrooms","area","zipcode","price"]
df = pd.read_csv("/content/drive/My Drive/Colab Notebooks/Houses Dataset/HousesInfo.txt" , sep=" ", header=None , names=cols)

df = df.loc[(df['price'] >= 100000) & (df['price'] <= 900000)]

# consider most expensive houses
df.sort_values(by = ['price'], ascending=False)

# consider top 200 most expensive houses
df = df.head(108)

# fetch those zipcodes
zipcodes = df['zipcode'].to_numpy()
zipcodes
```

Out[122]:
```
array([85255, 36372, 85266, 85262, 85266, 85266, 85266, 85266, 85255,
       85331, 85255, 85255, 85266, 93446, 93446, 85255, 93446, 98021,
       98021, 98021, 98021, 81524, 81524, 81524, 81524, 81418, 81418,
       81521, 81524, 81524, 81524, 81524, 81524, 81524, 81524, 62214,
       62234, 62034, 62025, 62025, 62088, 62234, 62234, 62234, 62234,
       62249, 62234, 62214, 62214, 60002, 60002, 62214, 60002, 60046,
       60016, 60016, 91901, 91901, 91901, 91901, 91901, 91901, 91901,
       91901, 91901, 91901, 91901, 91901, 91901, 93446, 91901, 91901,
       91901, 91901, 91901, 91901, 91901, 93446, 91901, 91901, 91901,
       91901, 92021, 92021, 92021, 92021, 92021, 92021, 92021, 92021,
       92021, 92021, 92677, 92677, 92677, 92677, 92677, 92677, 92677,
       92677, 92677, 92677, 92677, 92677, 92677, 92677, 92677, 96019])
```

In [123]:
```
df_x = df.drop(['price'], axis = 1)
df_y = df.loc[:, ['price']]

encode_numeric_zscore(df_x, 'Bedrooms')
encode_numeric_zscore(df_x, 'Bathrooms')
encode_numeric_zscore(df_x, 'area')
encode_text_dummy(df_x, 'zipcode')

df_x
```

Out[123]:

	Bedrooms	Bathrooms	area	zipcode-36372	zipcode-60002	zipcode-60016	zipcode-60046	zipcode-62025	zipcode-62034
0	0.585347	1.627947	1.001372	0	0	0	0	0	0
1	0.585347	0.389790	0.496395	1	0	0	0	0	0
2	-0.564061	1.627947	0.908911	0	0	0	0	0	0
6	-0.564061	1.627947	-0.071881	0	0	0	0	0	0
10	1.734755	2.866104	1.553289	0	0	0	0	0	0
...	...	...	...	...	...	...	...	...	...
142	-0.564061	-0.848367	-0.954522	0	0	0	0	0	0
144	-0.564061	-0.229288	-0.358508	0	0	0	0	0	0
145	-0.564061	-0.229288	-0.704168	0	0	0	0	0	0
147	-0.564061	-0.229288	-0.369888	0	0	0	0	0	0
151	-0.564061	-0.229288	-0.595349	0	0	0	0	0	0

108 rows × 26 columns

In [124]:
```
# drop corresponding rows from image dataset
img_arr = np.delete(img_arr, np.s_[108:], axis = 0)
img_arr.shape
```

Out[124]:
```
(108, 128, 128, 3)
```

In [126]:
```
x_train, x_test, y_train, y_test = train_test_split(df_x, df_y, test_size=0.3,
random_state=42)

#image data
img_train, img_test = train_test_split(img_arr, test_size = 0.3, random_state
= 42)

visible2 =  Input(shape=img_train.shape[1:])
conv1 = Conv2D(64, kernel_size = (3, 3), strides=(1, 1), activation='relu', pa
dding = 'same')(visible2)
pool1 = MaxPooling2D(pool_size = (2, 2))(conv1)
conv2 = Conv2D(32, kernel_size = (1,5), strides = (1,1), activation='relu')(po
ol1)
pool2 = MaxPooling2D(pool_size = (2, 2))(conv2)

visible1 = Input(shape=(x_train.shape[1],))
hidden1 = Dense(128, activation = 'relu')(visible1)
hidden2 = Dense(64, activation = 'relu')(hidden1)
dense_output = Dense(20, activation='relu')(hidden2)

#hidden2 = Dense(32, activation = 'relu')(hidden1)
#hidden3 = Dense(10, activation='relu')(hidden2)
#dense_output = Dense(1, activation='relu')(hidden3)
#dense_output = Dense(20, activation = 'relu')(hidden2)

# cnn
flat = Flatten()(pool2)
# nn
merge = concatenate([dense_output, flat])

# Final layers
final1 = Dense(16, activation='relu')(merge)
output = Dense(1)(final1)

model = Model(inputs = [visible1, visible2], outputs = output)
model.compile(loss='mean_squared_error', optimizer='adam')

checkpointer = ModelCheckpoint(filepath = "best_weights.hdf5", verbose = 0, sa
ve_best_only = True)
monitor = EarlyStopping(monitor = 'val_loss', min_delta = 1e-3, patience = 5,
verbose = 1, mode = 'auto')
model.fit([x_train, img_train], y_train, validation_data = ([x_test, img_tes
t], y_test), callbacks = [monitor, checkpointer], verbose = 2, epochs = 100)

model.load_weights('best_weights.hdf5')

# predict
pred = model.predict([x_test, img_test])
print(pred)

pred = np.asarray(pred)
y_test_arr = np.asarray(y_test)

# rmse
rmse = np.sqrt(metrics.mean_squared_error(pred, y_test_arr))
print("RMSE: {}".format(rmse))
```

```
# r2 score
_r2_score = metrics.r2_score(y_test_arr, pred)
print("R2 Score: {}".format(_r2_score))
```

```
Train on 75 samples, validate on 33 samples
Epoch 1/100
 - 3s - loss: 309422836066.9866 - val_loss: 375956489681.4545
Epoch 2/100
 - 2s - loss: 300905712407.8934 - val_loss: 357200329759.0303
Epoch 3/100
 - 2s - loss: 282356553700.6933 - val_loss: 322498277996.6061
Epoch 4/100
 - 2s - loss: 249338208583.6800 - val_loss: 265326412272.4849
Epoch 5/100
 - 2s - loss: 198553010612.9066 - val_loss: 182593584159.0303
Epoch 6/100
 - 2s - loss: 128602197633.7067 - val_loss: 89757101707.6364
Epoch 7/100
 - 2s - loss: 65387977495.8933 - val_loss: 44859912564.3636
Epoch 8/100
 - 2s - loss: 54938357596.1600 - val_loss: 66747598122.6667
Epoch 9/100
 - 2s - loss: 77615043706.8800 - val_loss: 54906328971.8788
Epoch 10/100
 - 2s - loss: 60394555583.1467 - val_loss: 45498877145.2121
Epoch 11/100
 - 2s - loss: 49047817530.0267 - val_loss: 56778135862.3030
Epoch 12/100
 - 2s - loss: 52731185015.4667 - val_loss: 61518343447.2727
Epoch 00012: early stopping
[[546603.5 ]
 [556871.1 ]
 [640142.1 ]
 [585761.5 ]
 [645646.44]
 [601847.5 ]
 [625730.2 ]
 [571604.1 ]
 [633155.44]
 [515529.28]
 [554395.25]
 [596083.44]
 [596343.  ]
 [626351.94]
 [592537.  ]
 [618604.25]
 [626553.6 ]
 [541063.9 ]
 [598178.2 ]
 [551634.94]
 [588143.44]
 [417922.78]
 [699061.8 ]
 [630930.75]
 [544830.6 ]
 [498092.28]
 [588223.56]
 [572755.6 ]
 [600699.3 ]
 [635208.4 ]
 [673164.8 ]
```

```
 [528877.8 ]
 [631620.3 ]]
RMSE: 211801.59044145016
R2 Score: 0.0028509080321448277
```

Additional feature 3: Comparing our best model with research paper

Our model	Emam's model
1.27873 * 10^5	2.79555 * 10^6

Thus, our model achieved a slightly better accuracy than Emam's model