LECTURE 7. CNN

(Convolutional Neural Network)

Part I. Image Pre-Processing

MANU 465

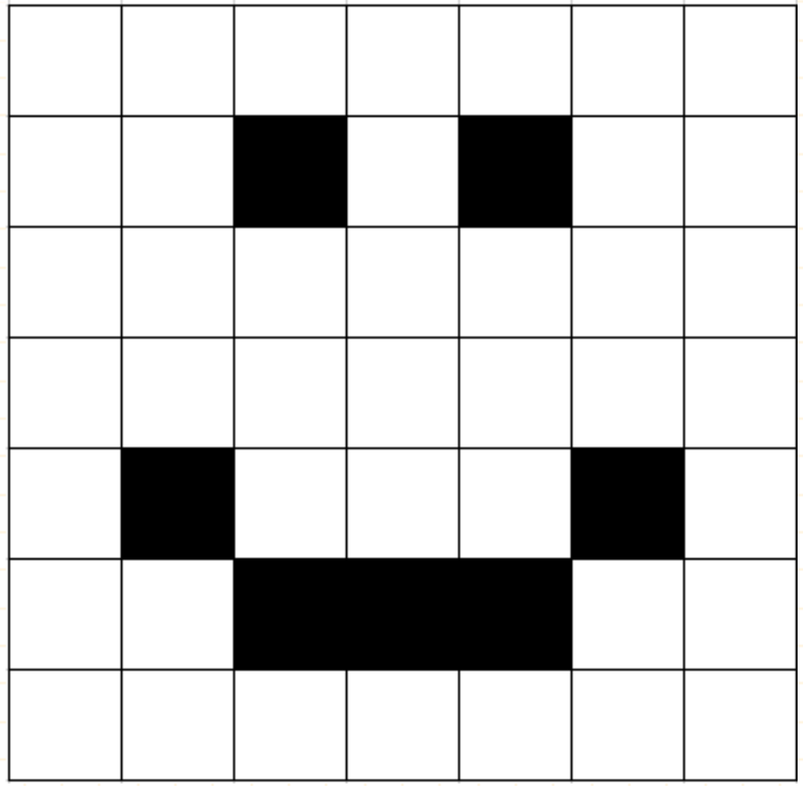
Ahmad Mohammadpanah

PhD, PEng

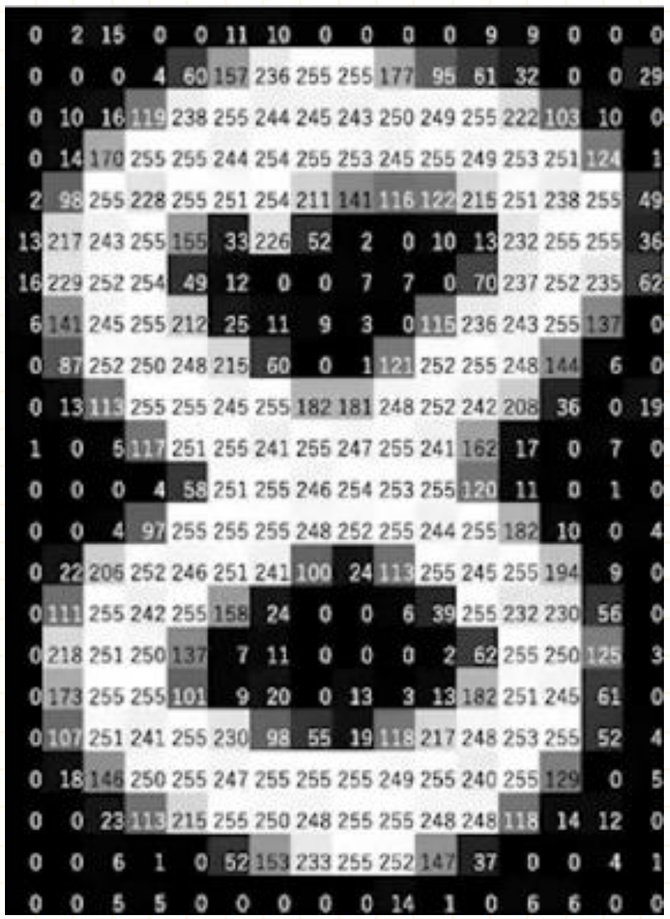
AIntelligentManufacturing.com

Image as a DataFrame

255	255	255	255	255	255	255
255	255	0	255	0	255	255
255	255	255	255	255	255	255
255	255	255	255	255	255	255
255	0	255	255	255	0	255
255	255	0	0	0	255	255
255	255	255	255	255	255	255

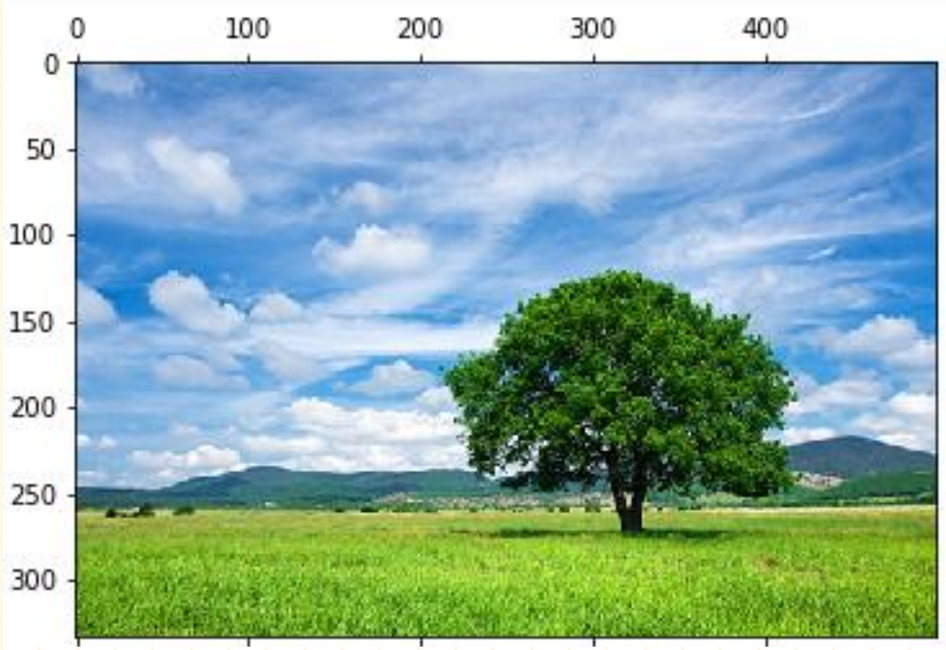


Higher Image Quality, Higher Number of Pixels, Larger DataFrame



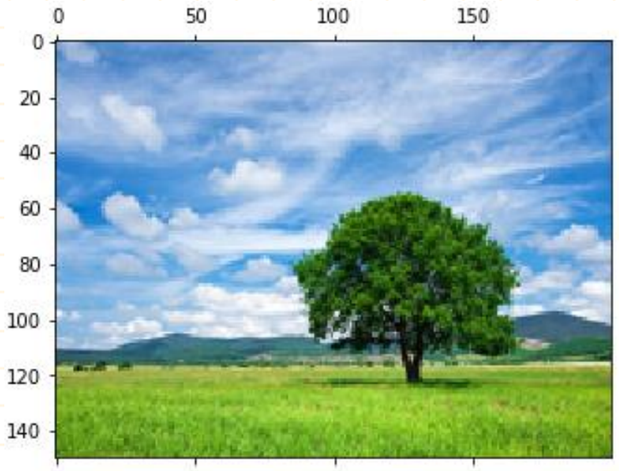
0	2	15	0	0	11	10	0	0	0	0	9	9	0	0	0
0	0	0	4	60	157	236	255	255	177	95	61	32	0	0	29
0	10	16	119	238	255	244	245	243	250	249	255	222	103	10	0
0	14	170	255	255	244	254	255	253	245	255	249	253	251	124	1
2	98	255	228	255	251	254	211	141	116	122	215	251	238	255	49
13	217	243	255	155	33	226	52	2	0	10	13	232	255	255	36
16	229	252	254	49	12	0	0	7	7	0	70	237	252	235	62
6	141	245	255	212	25	11	9	3	0	115	236	243	255	137	0
0	87	252	250	248	215	60	0	1	121	252	255	248	144	6	0
0	13	113	255	255	245	255	182	181	248	252	242	208	36	0	19
1	0	5	117	251	255	241	255	247	255	241	162	17	0	7	0
0	0	0	4	58	251	255	246	254	253	255	120	11	0	1	0
0	0	4	97	255	255	255	248	252	255	244	255	182	10	0	4
0	22	206	252	246	251	241	100	24	113	255	245	255	194	9	0
0	111	255	242	255	158	24	0	0	6	39	255	232	230	56	0
0	218	251	250	137	7	11	0	0	0	2	62	255	250	125	3
0	173	255	255	101	9	20	0	13	3	13	182	251	245	61	0
0	107	251	241	255	230	98	55	19	118	217	248	253	255	52	4
0	18	146	250	255	247	255	255	255	249	255	240	255	129	0	5
0	0	23	113	215	255	250	248	255	255	248	248	118	14	12	0
0	0	6	1	0	52	153	233	255	252	147	37	0	0	4	1
0	0	5	5	0	0	0	0	0	14	1	0	6	6	0	0

Image Resizing



	0	1	2	...	497	498	499
0	135	136	141	...	68	69	69
1	124	128	134	...	66	67	67
2	114	117	122	...	65	65	65
3	112	112	116	...	64	66	67
4	113	113	116	...	65	68	71
...	...	...	...	...	...	...	...
329	65	81	64	...	104	100	104
330	67	55	60	...	82	106	132
331	49	98	102	...	31	85	97
332	63	121	131	...	44	96	80
333	99	84	103	...	71	93	83

334 rows × 500 colun

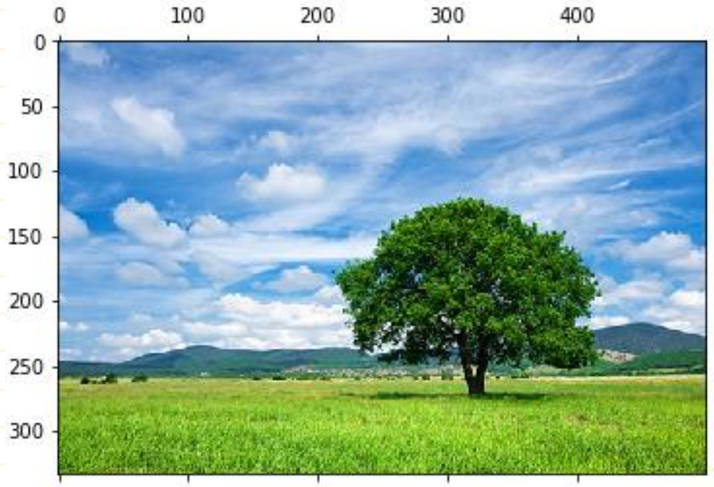


	0	1	2	3	...	196	197	198	199
0	129	146	165	179	...	62	62	65	67
1	113	126	144	163	...	59	59	62	66
2	121	128	134	148	...	66	66	66	71
3	133	127	129	140	...	75	73	70	72
4	107	103	110	133	...	75	72	72	73
...	...	...	...	...	...	...	...	...	...
145	92	94	83	95	...	103	117	108	90
146	105	91	91	78	...	91	86	90	98
147	83	67	87	65	...	88	81	82	85
148	70	71	100	100	...	95	76	85	97
149	97	103	100	84	...	75	77	66	79

150 rows × 200 columns

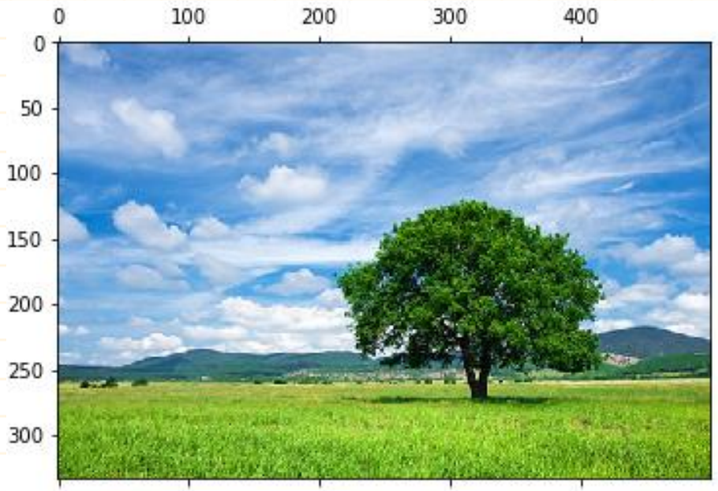
Image Scaling

We can also Scale (Normalize the Pixel Values) by dividing the DataFrame by 255



	0	1	2	...	497	498	499
0	135	136	141	...	68	69	69
1	124	128	134	...	66	67	67
2	114	117	122	...	65	65	65
3	112	112	116	...	64	66	67
4	113	113	116	...	65	68	71
...	...	...	...	...	...	...	...
329	65	81	64	...	104	100	104
330	67	55	60	...	82	106	132
331	49	98	102	...	31	85	97
332	63	121	131	...	44	96	80
333	99	84	103	...	71	93	83

334 rows × 500 colun

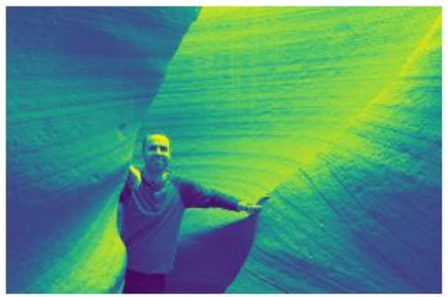


	0	1	2	...	497	498	499
0	0.529412	0.533333	0.552941	...	0.266667	0.270588	0.270588
1	0.486275	0.501961	0.525490	...	0.258824	0.262745	0.262745
2	0.447059	0.458824	0.478431	...	0.254902	0.254902	0.254902
3	0.439216	0.439216	0.454902	...	0.250980	0.258824	0.262745
4	0.443137	0.443137	0.454902	...	0.254902	0.266667	0.278431
...	...	...	...	...	...	...	...
329	0.254902	0.317647	0.250980	...	0.407843	0.392157	0.407843
330	0.262745	0.215686	0.235294	...	0.321569	0.415686	0.517647
331	0.192157	0.384314	0.400000	...	0.121569	0.333333	0.380392
332	0.247059	0.474510	0.513725	...	0.172549	0.376471	0.313725
333	0.388235	0.329412	0.403922	...	0.278431	0.364706	0.325490

334 rows × 500 columns

Colored Images, are composed of three primary colors, Red, Green, and Blue (RGB)

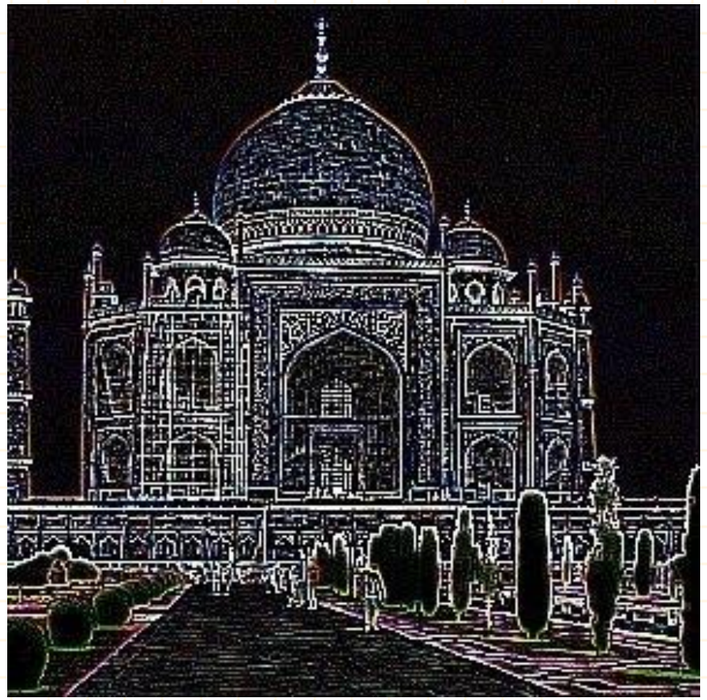
Each layer is a DataFrame,



	0	1	2	3	4	5	...	1356	1357	1358	1359	1360
0	4.0	6.0	6.0	5.0	3.0	4.0	...	85.0	88.0	107.0	108.0	125.0
1	4.0	6.0	6.0	3.0	4.0	8.0	...	90.0	109.0	119.0	122.0	131.0
2	6.0	3.0	3.0	5.0	5.0	8.0	...	105.0	124.0	123.0	126.0	122.0
3	3.0	3.0	2.0	3.0	7.0	6.0	...	117.0	121.0	125.0	126.0	126.0
4	3.0	1.0	4.0	3.0	4.0	7.0	...	128.0	129.0	138.0	132.0	131.0
...	...	...	...	...	...	...	...	...	...	...	...	...
898	6.0	9.0	7.0	8.0	7.0	2.0	...	2.0	2.0	4.0	3.0	6.0
899	6.0	8.0	8.0	8.0	8.0	4.0	...	4.0	4.0	4.0	2.0	4.0
900	8.0	7.0	9.0	6.0	8.0	8.0	...	5.0	5.0	2.0	1.0	3.0
901	4.0	5.0	10.0	7.0	8.0	12.0	...	4.0	4.0	1.0	2.0	2.0
902	2.0	5.0	7.0	6.0	8.0	13.0	...	2.0	2.0	4.0	4.0	4.0

Image Filtering

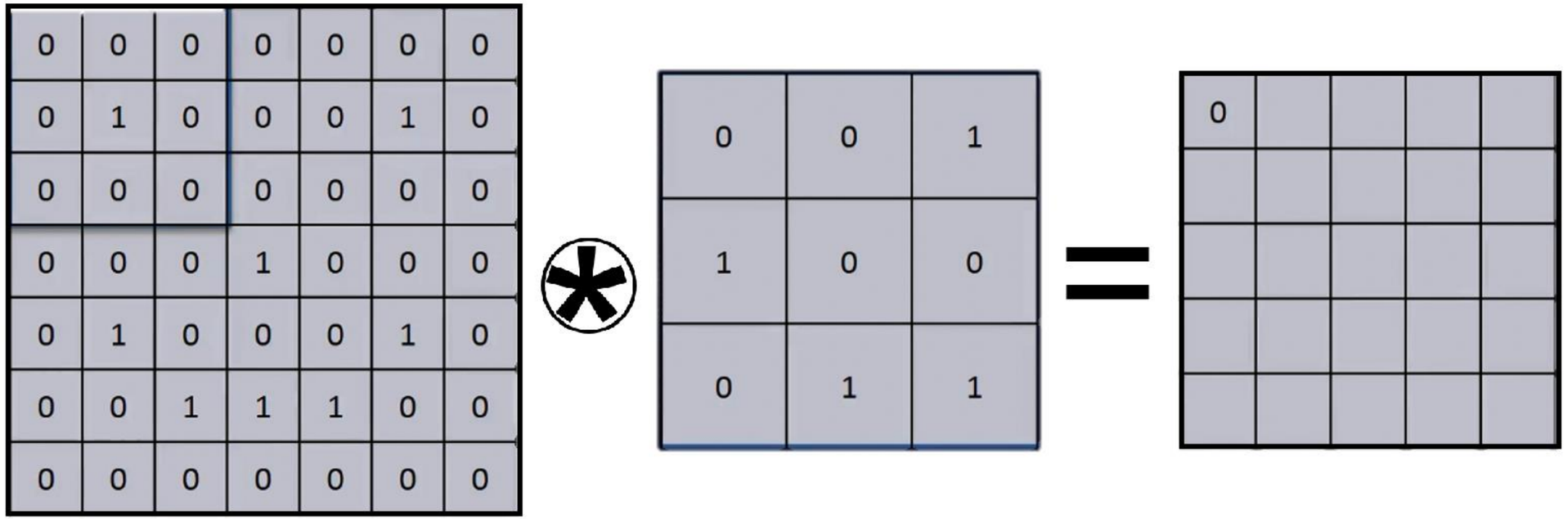
We can filter an Image to extract its different features



```
[[200., 201., 201., ..., 205., 214., 241.],
 [129., 131., 131., ..., 142., 161., 231.],
 [ 87.,  90.,  89., ..., 107., 130., 224.],
 [211., 211., 211., ..., 214., 221., 246.],
 [149., 151., 151., ..., 157., 174., 241.],
 [118., 120., 120., ..., 127., 147., 238.],
 ...,
 [  0.,   2.,   2., ..., 121., 118., 230.],
 [  0.,   1.,   1., ...,  70.,  95., 229.],
 [  0.,   1.,   1., ...,  28.,  67., 224.]]
```

```
[[  0.   0.   0. ...   0.   0.   0.]
 [  0. 200. 201. ... 214. 241.   0.]
 [  0. 129. 131. ... 161. 231.   0.]
 ...
 [  0.   0.   1. ...  95. 229.   0.]
 [  0.   0.   1. ...  67. 224.   0.]
 [  0.   0.   0. ...   0.   0.   0.]]
[[  0.   0.   0. ...   0.   0.   0.]
 [  0. 211. 211. ... 221. 246.   0.]
 [  0. 149. 151. ... 174. 241.   0.]
```

Convolution Process



For example this Filter (or, the technical term, "Kernel") emboss the image

0	0	0	0	0
0	-2	-1	0	0
0	-1	1	1	0
0	0	1	2	0
0	0	0	0	0

We need to only keep the important information in an image; for example, in these images, "the cat"

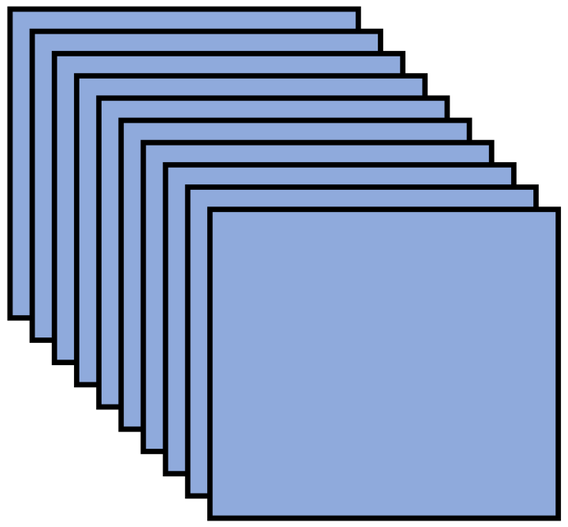
Max Pooling

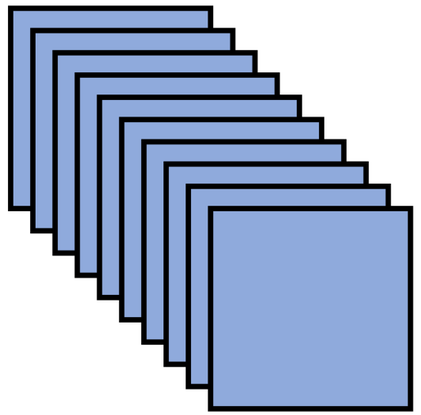
0	1	0	0	0
0	1	1	1	0
1	0	1	2	1
1	4	2	1	0
0	0	1	2	1

Flattening

1	1	1	1
1	1	2	2
4	4	2	2
4	4	2	2

Putting All Together





Summary

1. Image Pre-Processing Steps
2. Scaling
3. Resizing
4. Feature Extraction
5. Max Pooling
6. Flattening

More Image Pre-Processing: https://auth0.com/blog/image-processing-in-python-with-pillow/