Top_5 vs. Top_1 Correctness

- MobileNetV2를 활용한 ImageNet 분류

```
import warnings
warnings.filterwarnings('ignore')
```

Import Packages

```
import keras
import numpy as np
```

I. Tensorflow Hub

- 사전 훈련된 MobileNetV2

```
import tensorflow_hub as hub

url = "https://tfhub.dev/google/tf2-preview/mobilenet_v2/classification/2"

MbNetV2 = keras.Sequential([hub.KerasLayer(handle = url,
                                           input_shape = (224, 224, 3),
                                           trainable = False)])
```

- Model Summary

```
MbNetV2.summary()
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
keras_layer (KerasLayer)     (None, 1001)              3540265   
=================================================================
Total params: 3,540,265
Trainable params: 0
Non-trainable params: 3,540,265
_________________________________________________________________
```

II. ImageNetV2 TopImages

1) 이미지(ImageNetV2) 다운로드

- 1000가지 객체에 대한 이미지 10장씩 구성
  - 약 2분

```
%%time

import os

image_url = 'https://s3-us-west-2.amazonaws.com/imagenetv2public/imagenetv2-top-images.tar.gz'
data_dir = '/content/'

data_root_orig = keras.utils.get_file('imagenetV2',
                                      image_url,
                                      cache_dir = data_dir,
                                      extract = True)
```

```
Downloading data from https://s3-us-west-2.amazonaws.com/imagenetv2public/imagenetv2-top-imag
1245929472/1245927936 [==============================] - 44s 0us/step
CPU times: user 9.25 s, sys: 6.46 s, total: 15.7 s
Wall time: 50.7 s
```

- 다운로드 경로 확인

```
import pathlib
from glob import glob

data_root = pathlib.Path(glob('/content/datasets/*/')[0])
print(data_root)
```

```
/content/datasets/imagenetv2-top-images-format-val
```

- 이미지별 디렉토리 확인
  - 1000개

```
!ls /content/datasets/imagenetv2-top-images-format-val
```

```
0    155  211  269  325  382  439  496  552  609  666  722  78   836  893  95
1    156  212  27   326  383  44   497  553  61   667  723  780  837  894  950
10   157  213  270  327  384  440  498  554  610  668  724  781  838  895  951
100  158  214  271  328  385  441  499  555  611  669  725  782  839  896  952
101  159  215  272  329  386  442  5    556  612  67   726  783  84   897  953
102  16   216  273  33   387  443  50   557  613  670  727  784  840  898  954
103  160  217  274  330  388  444  500  558  614  671  728  785  841  899  955
104  161  218  275  331  389  445  501  559  615  672  729  786  842  9    956
105  162  219  276  332  39   446  502  56   616  673  73   787  843  90   957
106  163  22   277  333  390  447  503  560  617  674  730  788  844  900  958
107  164  220  278  334  391  448  504  561  618  675  731  789  845  901  959
108  165  221  279  335  392  449  505  562  619  676  732  79   846  902  96
109  166  222  28   336  393  45   506  563  62   677  733  790  847  903  960
11   167  223  280  337  394  450  507  564  620  678  734  791  848  904  961
```

110	168	224	281	338	395	451	508	565	621	679	735	792	849	905	962
111	169	225	282	339	396	452	509	566	622	68	736	793	85	906	963
112	17	226	283	34	397	453	51	567	623	680	737	794	850	907	964
113	170	227	284	340	398	454	510	568	624	681	738	795	851	908	965
114	171	228	285	341	399	455	511	569	625	682	739	796	852	909	966
115	172	229	286	342	4	456	512	57	626	683	74	797	853	91	967
116	173	23	287	343	40	457	513	570	627	684	740	798	854	910	968
117	174	230	288	344	400	458	514	571	628	685	741	799	855	911	969
118	175	231	289	345	401	459	515	572	629	686	742	8	856	912	97
119	176	232	29	346	402	46	516	573	63	687	743	80	857	913	970
12	177	233	290	347	403	460	517	574	630	688	744	800	858	914	971
120	178	234	291	348	404	461	518	575	631	689	745	801	859	915	972
121	179	235	292	349	405	462	519	576	632	69	746	802	86	916	973
122	18	236	293	35	406	463	52	577	633	690	747	803	860	917	974
123	180	237	294	350	407	464	520	578	634	691	748	804	861	918	975
124	181	238	295	351	408	465	521	579	635	692	749	805	862	919	976
125	182	239	296	352	409	466	522	58	636	693	75	806	863	92	977
126	183	24	297	353	41	467	523	580	637	694	750	807	864	920	978
127	184	240	298	354	410	468	524	581	638	695	751	808	865	921	979
128	185	241	299	355	411	469	525	582	639	696	752	809	866	922	98
129	186	242	3	356	412	47	526	583	64	697	753	81	867	923	980
13	187	243	30	357	413	470	527	584	640	698	754	810	868	924	981
130	188	244	300	358	414	471	528	585	641	699	755	811	869	925	982
131	189	245	301	359	415	472	529	586	642	7	756	812	87	926	983
132	19	246	302	36	416	473	53	587	643	70	757	813	870	927	984
133	190	247	303	360	417	474	530	588	644	700	758	814	871	928	985
134	191	248	304	361	418	475	531	589	645	701	759	815	872	929	986
135	192	249	305	362	419	476	532	59	646	702	76	816	873	93	987
136	193	25	306	363	42	477	533	590	647	703	760	817	874	930	988
137	194	250	307	364	420	478	534	591	648	704	761	818	875	931	989
138	195	251	308	365	421	479	535	592	649	705	762	819	876	932	99
139	196	252	309	366	422	48	536	593	65	706	763	82	877	933	990
14	197	253	31	367	423	480	537	594	650	707	764	820	878	934	991
140	198	254	310	368	424	481	538	595	651	708	765	821	879	935	992
141	199	255	311	369	425	482	539	596	652	709	766	822	88	936	993
142	2	256	312	37	426	483	54	597	653	71	767	823	880	937	994
143	20	257	313	370	427	484	540	598	654	710	768	824	881	938	995
144	200	258	314	371	428	485	541	599	655	711	769	825	882	939	996
145	201	259	315	372	429	486	542	6	656	712	77	826	883	94	997
146	202	26	316	373	43	487	543	60	657	713	770	827	884	940	998
147	203	260	317	374	430	488	544	600	658	714	771	828	885	941	999
148	204	261	318	375	431	489	545	601	659	715	772	829	886	942	
149	205	262	319	376	432	49	546	602	66	716	773	83	887	943	
15	206	263	32	377	433	490	547	603	660	717	774	830	888	944	
150	207	264	320	378	434	491	548	604	661	718	775	831	889	945	

2) Image Label 다운로드

- 'ImageNetLabels.txt'

```
import pandas as pd

url = 'https://storage.googleapis.com/download.tensorflow.org/data/ImageNetLabels.txt'
DF = pd.read_table(url, header = None)

label_text = DF.values.reshape(-1)
```

- Label 확인
  - 1001개 주의!!!

```
print(len(label_text))
print(label_text[:5])
print(label_text[-5:])
```

```
1001
['background' 'tench' 'goldfish' 'great white shark' 'tiger shark']
['earthstar' 'hen-of-the-woods' 'bolete' 'ear' 'toilet tissue']
```

3) Image(X) vs. Label(y) 확인

- 이미지 랜덤 추출

```
import random

all_image_paths = list(data_root.glob('*/*'))
all_image_paths = [str(path) for path in all_image_paths]

random.choice(all_image_paths)
```

```
'/content/datasets/imagenetv2-top-images-format-val/839/3f0f9990baa3f389cd85ef85e9c5d6dde02c7
```

- 9개의 Image vs. Label 출력

```
import PIL.Image as Image
import matplotlib.pyplot as plt

random.shuffle(all_image_paths)

image_count = len(all_image_paths)
print('image_count:', image_count)

plt.figure(figsize = (12, 12))
for n in range(9):
    image_path = random.choice(all_image_paths)
    plt.subplot(3, 3 , n + 1)
    plt.imshow(plt.imread(image_path))
    idx = int(image_path.split('/')[-2]) + 1
    plt.title(str(idx) + ' : ' + label_text[idx])
    plt.axis('off')
plt.show()
```

III. MobileNet의 분류 라벨 확인

```
import cv2
```

```
plt.figure(figsize = (16, 16))

def softmax(x):
    m = np.max(x)
    sm = np.exp(x - m)/np.sum(np.exp(x - m))
    return sm

for n in range(3):
    # 랜덤 이미지 경로 추출
    image_path = random.choice(all_image_paths)

    # 이미지 출력
    plt.subplot(3, 2, n * 2 + 1)
    plt.imshow(plt.imread(image_path))
    idx = int(image_path.split('/')[-2]) + 1
    plt.title(str(idx) + ' : ' + label_text[idx])
    plt.axis('off')

    # Top-5 예측값 출력 공간
    plt.subplot(3, 2 , n * 2 + 2)

    # 이미지 크기 변경 및 정규화
    img = cv2.imread(image_path)
    img = cv2.resize(img, dsize = (224, 224))
    img = img / 255.0
    img = np.expand_dims(img, axis = 0)

    # MobileNetV2 이미지 분류
    logits = MbNetV2.predict(img)[0]
    prediction = softmax(logits)

    # Top-5 예측값
    top_5_predict = prediction.argsort()[::-1][:5]
    labels = [label_text[index] for index in top_5_predict]

    # Top-5 막대그래프 색상 지정
    color = ['gray'] * 5
    if idx in top_5_predict:
        color[top_5_predict.tolist().index(idx)] = 'green'
    color = color[::-1]

    # Top-5 막대그래프 출력
    plt.barh(range(5), prediction[top_5_predict][::-1] * 100, color = color)
    plt.yticks(range(5), labels[::-1])
```

- np.argsort( )
    - 오름차순 정렬 후 Index 반환

```
AR = np.array([5555, 55, 5, 555])
print('오름차순 정렬 :', np.sort(AR), 'Wn')

arg = np.argsort(AR)
```

```
print('오름차순 정렬 후 Index :', arg, '₩n')

print('argsort 결과 적용 :', AR[arg])
```

```
오름차순 정렬 : [   5   55  555 5555]

오름차순 정렬 후 Index : [2 1 3 0]

argsort 결과 적용 : [   5   55  555 5555]
```

IV. MobileNetV2의 분류 성능 확인

- 약 9분

```
%%time

import cv2

top_1 = 0
top_5 = 0

for image_path in all_image_paths:
    img = cv2.imread(image_path)
    img = cv2.resize(img, dsize = (224, 224))
    img = img / 255.0
    img = np.expand_dims(img, axis = 0)

    top_5_predict = MbNetV2.predict(img)[0].argsort()[::-1][:5]
    idx = int(image_path.split('/')[-2]) + 1

    if idx in top_5_predict:
        top_5 =  top_5 + 1
        if top_5_predict[0] == idx:
            top_1 = top_1 + 1

print('Top-5 Correctness:', top_5 / len(all_image_paths) * 100, '%')
print('Top-1 Correctness:', top_1 / len(all_image_paths) * 100, '%')
```

```
Top-5 Correctness: 83.52000000000001 %
Top-1 Correctness: 59.06 %
CPU times: user 8min 17s, sys: 17.3 s, total: 8min 34s
Wall time: 8min 17s
```

#

#

#

The End

#

#

#