Ambient AI Bootcamp
Practice 3

SNU Graduate School of Data Science

Contents

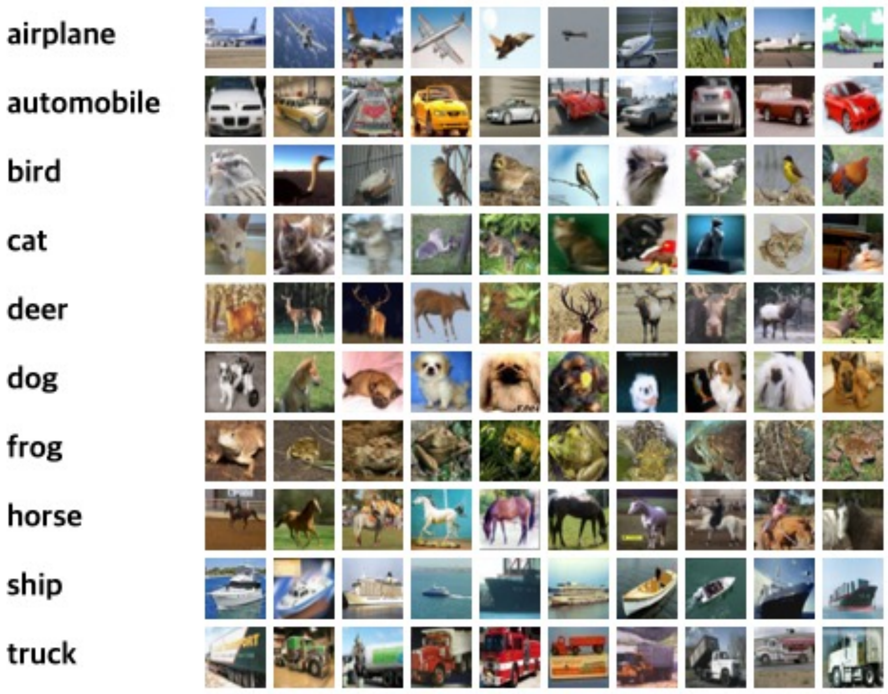
- Lightweight CNN for 2D object classification
- Transfer Learning
- HW3: MobileNet v3

3-1. Lightweight CNN for 2D classification

0. Dataset

Dataset: Cifar-10

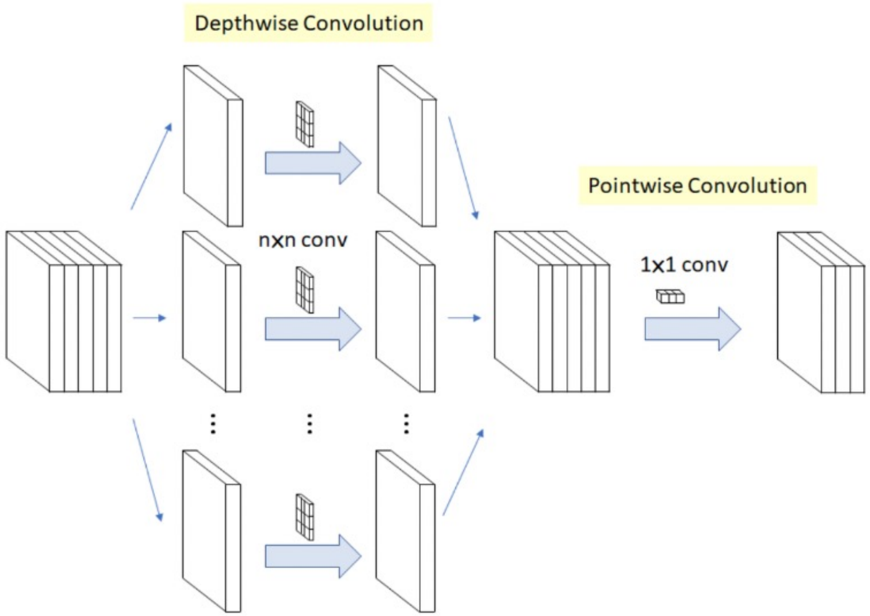
- 10개 class의 32x32 이미지
- 각 class별 6,000장씩 총 60,000장
- 50,000개의 training set과 10,000개의 test set으로 이루어져 있음



1. MobileNetV2

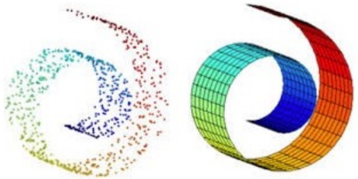
MobileNetV2: Inverted Residuals and Linear Bottlenecks, *CVPR* (2018)

- 임베디드 디바이스 또는 모바일 장치를 타겟으로하는 **단순한 구조의 경량화 네트워크**
- **MobileNetV1을 개선함**
  - **Depthwise-Separable Convolution**을 단순히 쌓지 않고, **Inverted Residual Block**을 도입함



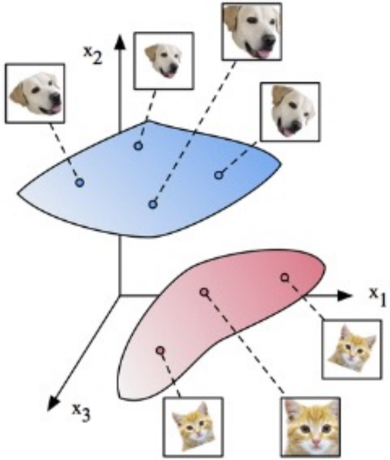
1. MobileNetV2

(1) Inverted Residual and Linear Bottleneck



Linear Bottlenecks

- **Manifold of Interest**
  - Manifold: high-dimensional 데이터를 low-dimension으로 옮길 때 데이터를 잘 설명하는 집합의 모형
  - 데이터가 Neural Network에 입력되면, 특정 레이어의 activation은 “manifold of interest”를 구성
  - “manifold of interest”는 low-dimensional subspace로 embedding 가능. 즉, 고차원 정보는 저차원에서 표현 가능 (가정)

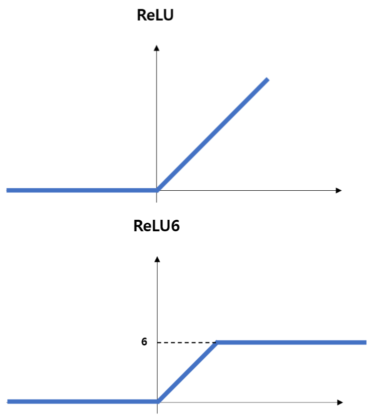


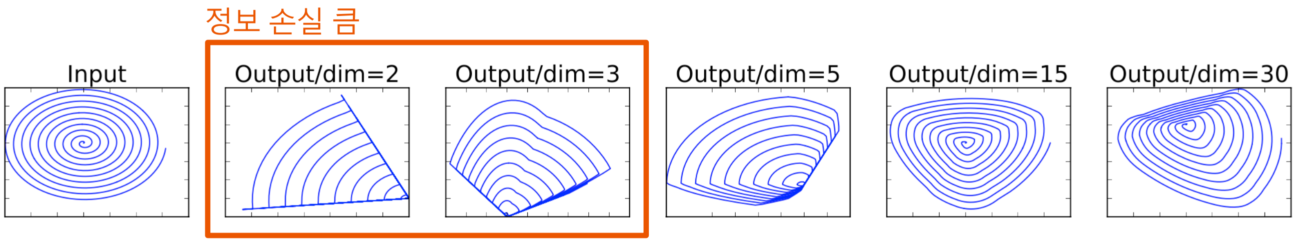
1. MobileNetV2

(1) Inverted Residual and Linear Bottleneck

Linear Bottlenecks

- **Linear Transformation**
  - Subspace로 embedding하는 과정에서 ReLU를 사용하면서 필연적으로 정보의 손실 발생 (“manifold of interest”가 ReLU 적용 후 non-zero volume을 가지면 linear transformation 된 것으로 정보 보존)
  - 많은 채널을 사용할 경우 정보 보존이 가능하다고 주장함





1. MobileNetV2

(1) Inverted Residual and Linear Bottleneck

Linear Bottlenecks

- **Linear Bottlenecks**
  - Input의 중요한 정보들인 “manifold of interest”는 layer를 거치며 low-dimensional 영역으로 전달될 수 있는데, 이때 layer가 linear transformation이면 정보가 보존된다고 가정할 수 있음
  - MobileNetV2에서는 low-dimension으로 mapping하는 linear transformation을 만들 때 bottleneck 구조를 사용함
  - 즉, linear transformation역할을 하는 linear bottleneck layer를 활용해서 dimension은 줄이고 중요한 정보는 그대로 유지하는 전략을 취함

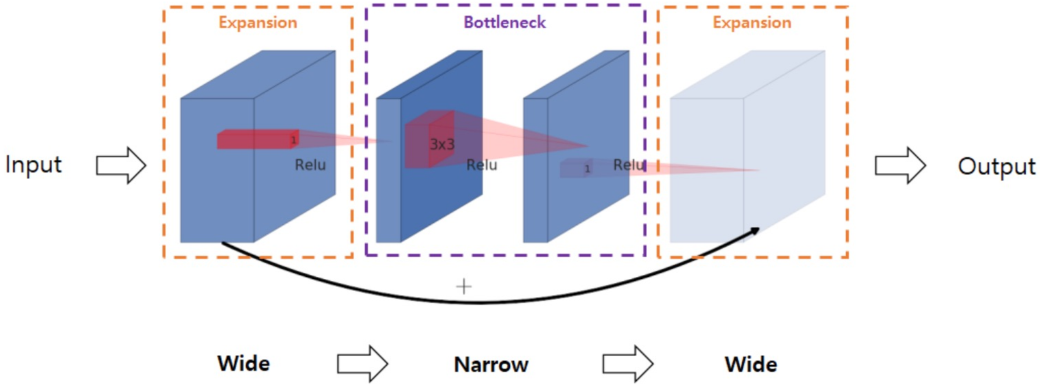
1. MobileNetV2

(1) Inverted Residual and Linear Bottleneck

Inverted Residuals

- 일반적인 Residual block은
  Wide– Narrow – Wide형태

Residual block



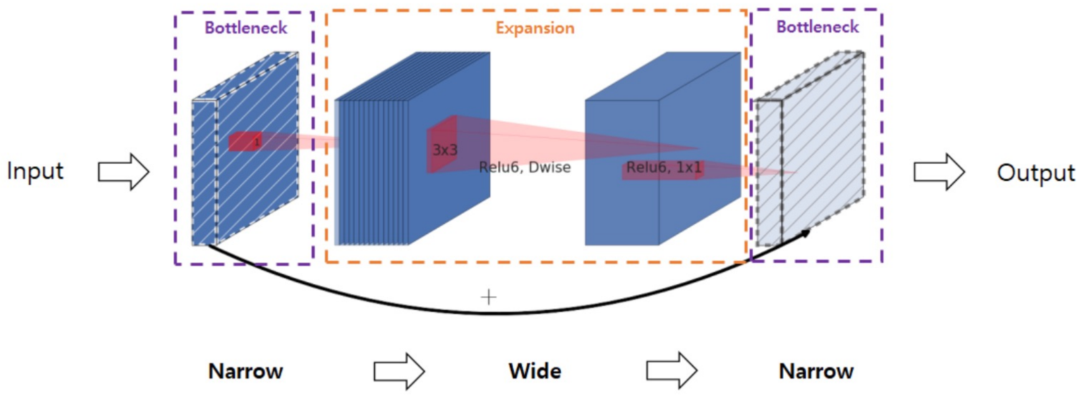
1. MobileNetV2

(1) Inverted Residual and Linear Bottleneck

Inverted Residuals

- 고차원의 정보는 저차원에 표현 가능하다고 가정했으므로 bottleneck block에 이미 “manifold of interest”가 보존되어 있다고 생각할 수 있음
- Shortcut connectio을 bottleneck block끼리 연결 (expansion에서는 non-linearity 담당)

Inverted residual block

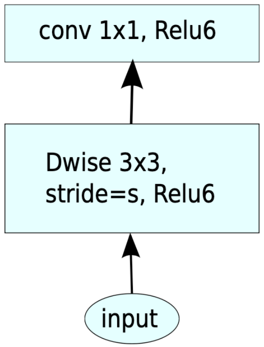


1. MobileNetV2

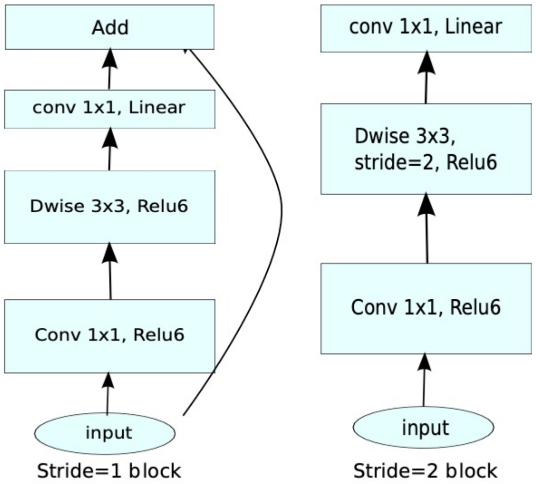
(1) Inverted Residual and Linear Bottleneck

Bottleneck Residual Block

MobileNetV1



MobileNetV2



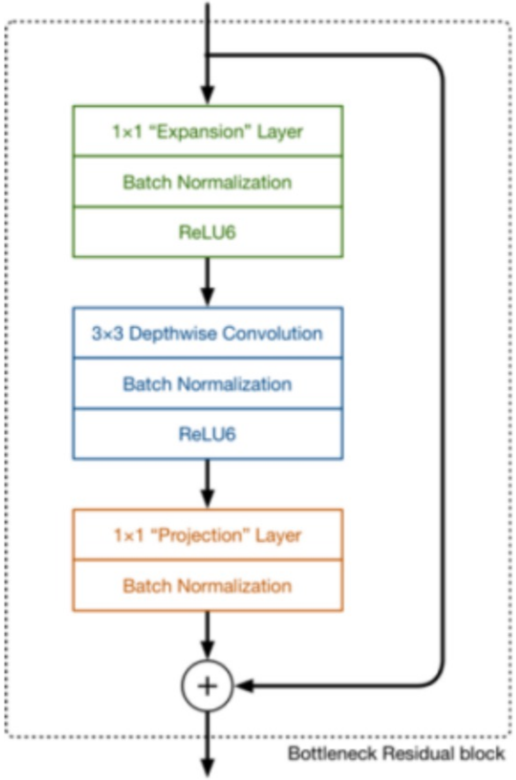
1. MobileNetV2

(1) Inverted Residual and Linear Bottleneck

Bottleneck Residual Block

Input	Operator	Output
$h \times w \times k$	1x1 conv2d , ReLU6	$h \times w \times (tk)$
$h \times w \times tk$	3x3 dwise s=s, ReLU6	$\frac{h}{s} \times \frac{w}{s} \times (tk)$
$\frac{h}{s} \times \frac{w}{s} \times tk$	linear 1x1 conv2d	$\frac{h}{s} \times \frac{w}{s} \times k'$

Table 1: *Bottleneck residual block* transforming from k to k' channels, with stride s, and expansion factor t.



1. MobileNetV2

(2) MobileNetV2 Network

Input	Operator	t	c	n	s
$224^2 \times 3$	conv2d	-	32	1	2
$112^2 \times 32$	bottleneck	1	16	1	1
$112^2 \times 16$	bottleneck	6	24	2	2
$56^2 \times 24$	bottleneck	6	32	3	2
$28^2 \times 32$	bottleneck	6	64	4	2
$14^2 \times 64$	bottleneck	6	96	3	1
$14^2 \times 96$	bottleneck	6	160	3	2
$7^2 \times 160$	bottleneck	6	320	1	1
$7^2 \times 320$	conv2d 1x1	-	1280	1	1
$7^2 \times 1280$	avgpool 7x7	-	-	1	-
$1 \times 1 \times 1280$	conv2d 1x1	-	k	-	

Table 2: MobileNetV2 : Each line describes a sequence of 1 or more identical (modulo stride) layers, repeated n times. All layers in the same sequence have the same number c of output channels. The first layer of each sequence has a stride s and all others use stride 1. All spatial convolutions use 3×3 kernels. The expansion factor t is always applied to the input size as described in Table 1.

Input	Operator	expansion	filters	strides	개수
32x32x3	3x3 Conv2D	-	32	2	1
16x16x32	inverted_res_block	1	16	1	1
16x16x16	inverted_res_block	6	24	2	2
8x8x24	inverted_res_block	6	32	2	3
4x4x32	inverted_res_block	6	64	2	4
2x2x64	inverted_res_block	6	96	1	3
2x2x96	inverted_res_block	6	160	2	3
1x1x160	inverted_res_block	6	320	1	1
1x1x320	1x1 Conv2D	-	1280	1	1
1x1x1280	GlobalAvgPool	-	-	-	1
1280	Dense	-	-	-	1

2. Training

- 30 epoch 이상 training
- Training 예시

```
Epoch 20/50
1563/1563 [==============================] - 64s 41ms/step - loss: 0.7281 - accuracy: 0.7926 - val_loss: 1.0559 - val_accuracy: 0.6847
Epoch 21/50
1563/1563 [==============================] - 64s 41ms/step - loss: 0.6938 - accuracy: 0.8020 - val_loss: 0.9425 - val_accuracy: 0.7249
Epoch 22/50
1563/1563 [==============================] - 65s 42ms/step - loss: 0.6753 - accuracy: 0.8087 - val_loss: 0.9696 - val_accuracy: 0.7139
Epoch 23/50
1563/1563 [==============================] - 65s 42ms/step - loss: 0.6575 - accuracy: 0.8179 - val_loss: 0.9245 - val_accuracy: 0.7381
Epoch 24/50
1563/1563 [==============================] - 65s 42ms/step - loss: 0.6608 - accuracy: 0.8184 - val_loss: 1.0376 - val_accuracy: 0.7124
Epoch 25/50
1563/1563 [==============================] - 66s 42ms/step - loss: 0.6559 - accuracy: 0.8204 - val_loss: 1.0913 - val_accuracy: 0.6995
Epoch 26/50
1563/1563 [==============================] - 65s 42ms/step - loss: 0.6393 - accuracy: 0.8259 - val_loss: 0.9902 - val_accuracy: 0.7170
Epoch 27/50
1563/1563 [==============================] - 63s 40ms/step - loss: 0.6155 - accuracy: 0.8362 - val_loss: 0.9250 - val_accuracy: 0.7432
Epoch 28/50
1563/1563 [==============================] - 63s 40ms/step - loss: 0.6083 - accuracy: 0.8394 - val_loss: 1.0261 - val_accuracy: 0.7174
Epoch 29/50
1563/1563 [==============================] - 66s 42ms/step - loss: 0.6549 - accuracy: 0.8233 - val_loss: 1.1048 - val_accuracy: 0.6968
Epoch 30/50
1563/1563 [==============================] - 67s 43ms/step - loss: 0.6497 - accuracy: 0.8261 - val_loss: 0.9958 - val_accuracy: 0.7233
```

3-2. Transfer Learning

Overview

ImageNet Pre-trained MobileNetV2를 이용한 Transfer Learning

- **ImageNet dataset**
  - 대표적인 대규모(large-scale) 데이터셋으로 1,000개의 클래스로 구성되며, 100만개가 넘는 데이터를 포함한다. (각 클래스당 약 1,000개의 사진으로 구성)
  - 개와 고양이도 종별로 세분화 되어있는데, Transfer Learning을 통해 **개와 고양이로 분류하는 Binary classification model**을 만든다.

```
'German_shepherd', 0.9796372),
'malinois', 0.020184083),
'Norwegian_elkhound', 0.00015799515),
'African_hunting_dog', 5.2901587e-06),
'briard', 3.9127376e-06)]]
```

```
standard_poodle', 0.5650911),
miniature_poodle', 0.37279922),
Irish_water_spaniel', 0.053150617),
toy_poodle', 0.0072146286),
Kerry_blue_terrier', 0.0013652634)]]
```

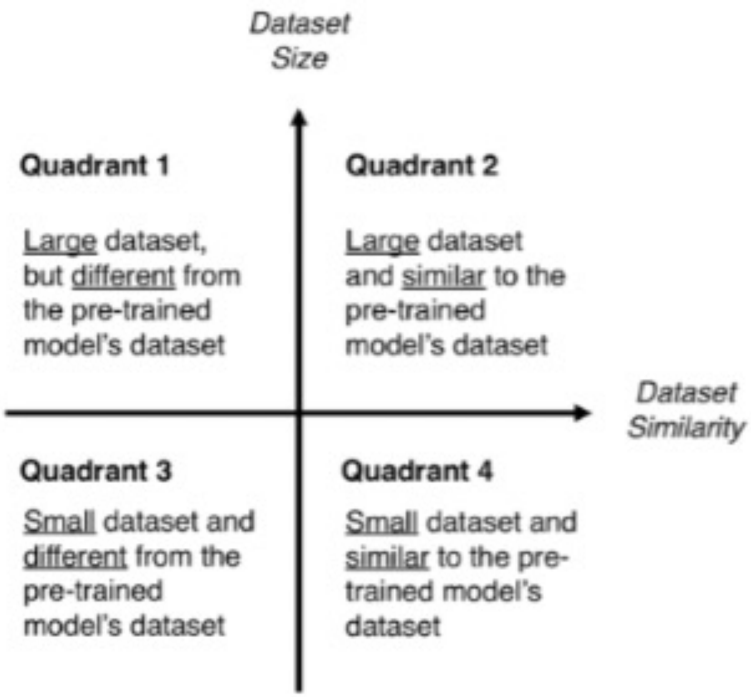
151	Chihuahua
152	Japanese spaniel
153	Maltese dog, Maltese terrier, Maltese
154	Pekinese, Pekingese, Peke
155	Shih-Tzu
156	Blenheim spaniel
157	papillon
158	toy terrier
159	Rhodesian ridgeback
160	Afghan hound, Afghan
161	basset, basset hound
162	beagle

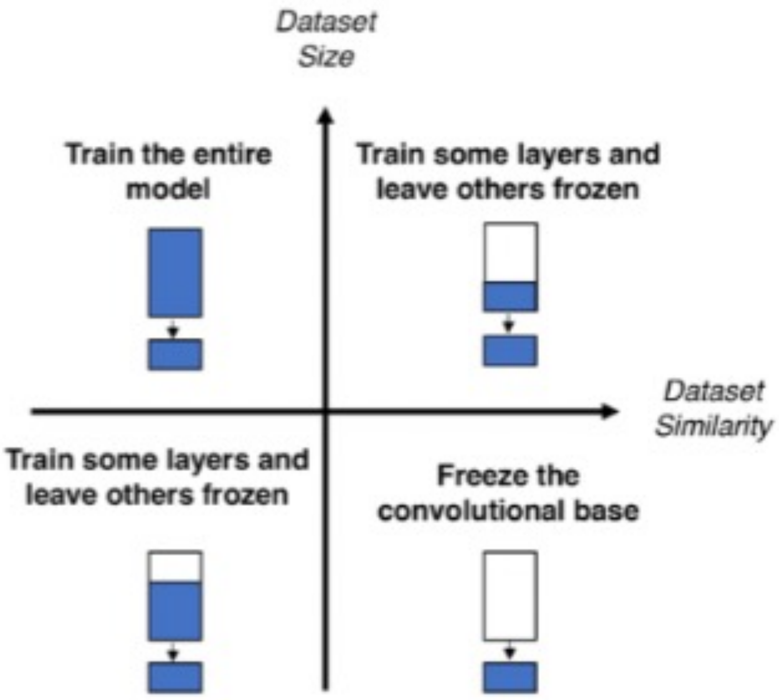
Dataset

Dog and Cat Dataset

- 개/고양이 이미지로만 구성된 데이터셋
- Train 2,000장, Validation 1,000장

Transfer Learning & Fine Tuning





Thanks!