Car Classifier

using Inception V3

* * * | * * * | * * *

목 차

차종 분류 시스템

- 고속도로 톨게이트나 일반도로에서 교통량 조사

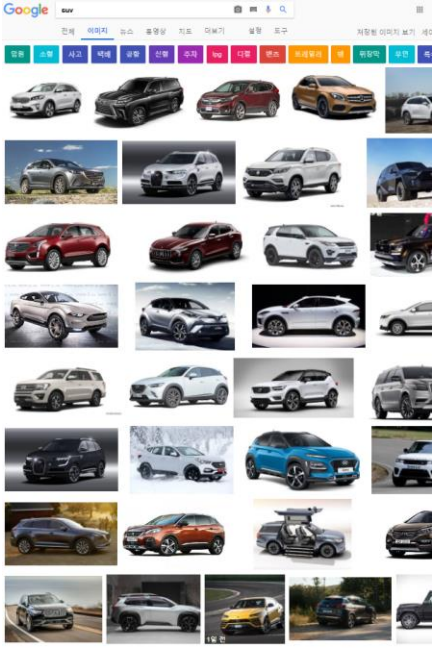
- 전체 교통량 중 각 차종이 차지하는 비율 통계

- 특정 도로 / 교각을 통과하는 차량 분석

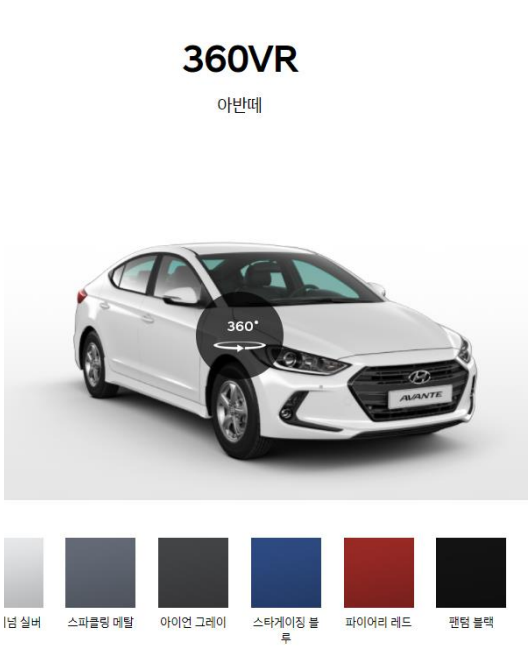
- 주차장 차량 비율 분석 및 개선

데이터 수집

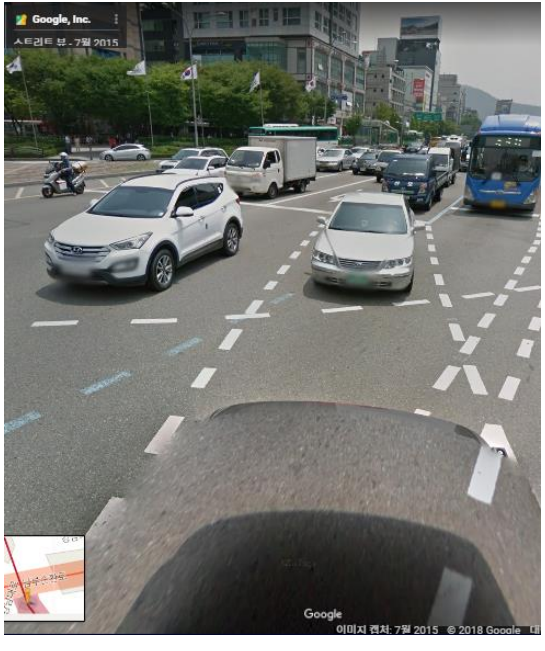
✓ 구글(crawling)



✓ 3D 모델(macro)



✓ 로드뷰



데이터 수집

- 다양한 방식을 사용
  - 구글링, 3D 모델, 로드뷰
- 다양한 색상 및 모델 수집
  - macro를 이용하여
- 수집한 자료 중 배경색에 따라서 정확도가 떨어지는 문제 발생
  - 구글링 데이터 위주로 훈련 (배경이 길거리, 도로인 것)
- 차량의 전면 범퍼를 포함하도록 수집

- Train-set
  - 각 차종 별 80개 씩
- Valid-set
  - 각 차종 별 25개 씩
  - 3D모델, 로드뷰를 섞어서

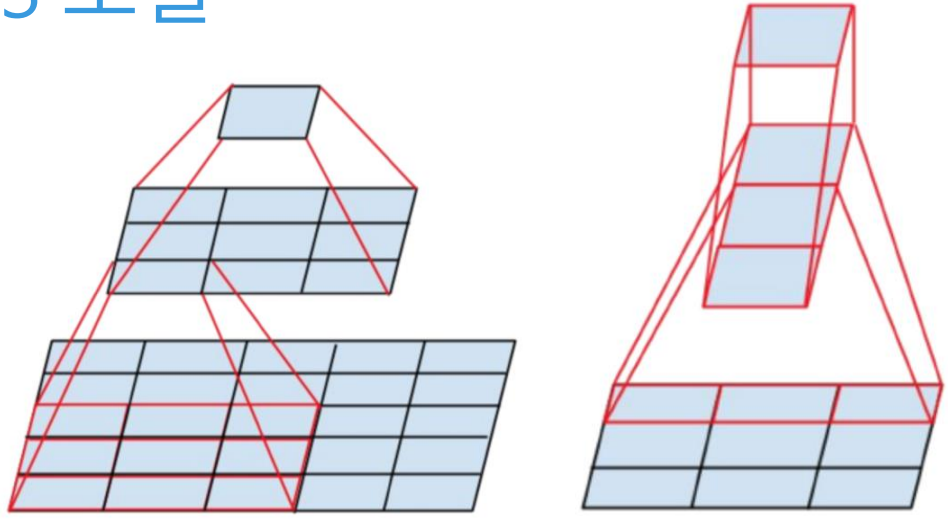
인셉션 V3 모듈

- Inception V1(GoogLeNet)을 개선한 모듈
- Neural-Net.의 디자인 원칙을 따름
- VGGNet과 GoogLeNet에 비해 개선된 성능을 보인다

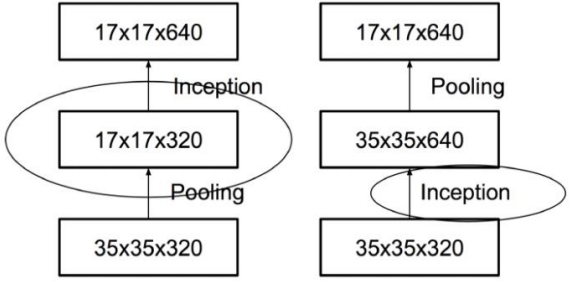
(1) Avoid representational bottlenecks, especially early in the network
(2) Higher dimensional representations are easier to process locally within a network.
(3) Spatial aggregation can be done over lower dimensinal embeddings without much or any loss in representational power
(4) Balance the width and depth of the network.

Network	Models Evaluated	Crops Evaluated	Top-1 Error	Top-5 Error
VGGNet [18]	2	-	23.7%	6.8%
GoogLeNet [20]	7	144	-	6.67%
PReLU [6]	-	-	-	4.94%
BN-Inception [7]	6	144	20.1%	4.9%
Inception-v3	4	144	**17.2%**	**3.58%** *

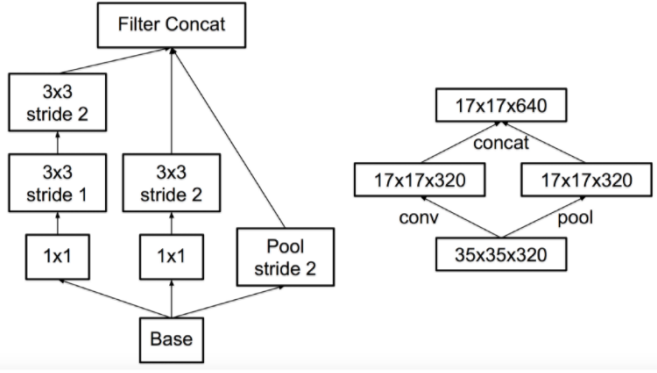
인셉션 V3 모델



- 3X3 conv 2개로 factorization
  - 5X5에 비해 연산량이 약 28% 줄어든다
- Asymmetric conv factorization
  - 3X3에 비해 연산량이 약 33% 줄어든다
- Inception과 Pooling을 병렬로 수행
  - 연산량이 줄어들고 Representation Bottleneck을 없앤다.

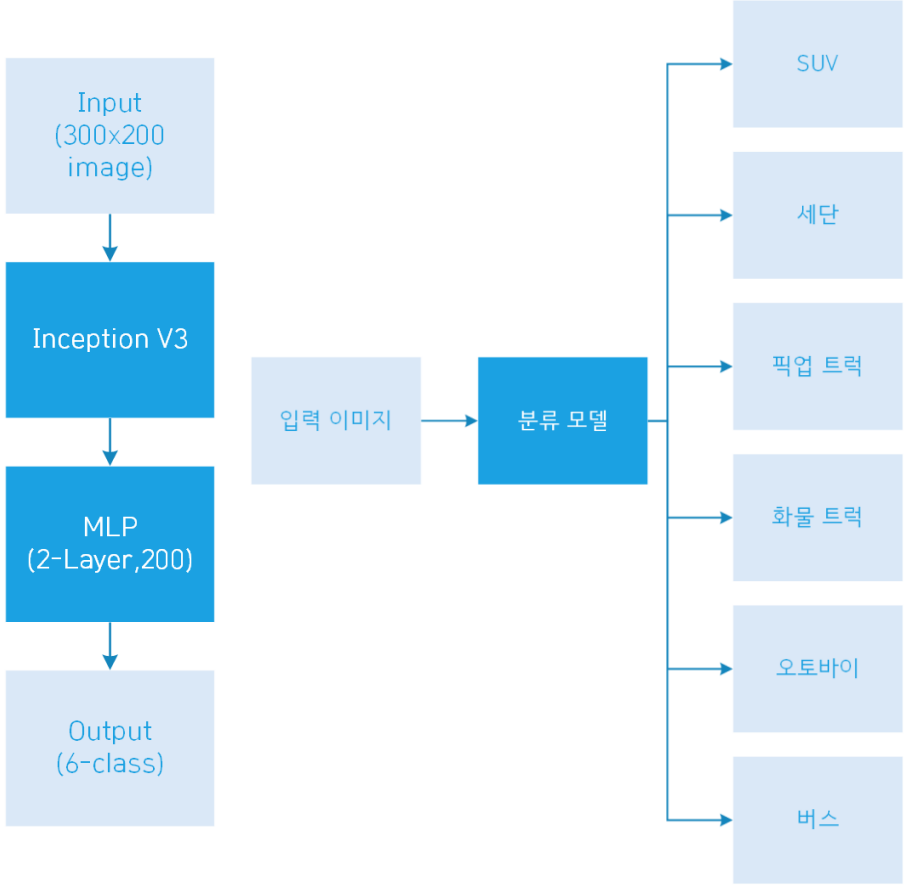


representational bottleneck　　expansive computation



초기 모델 구조

- ✓ 차량 이미지 처리
  - 인셉션 모듈 이용
  - MLP load(x)
- ✓ 분류
  - Multi Layer Perceptron
- ✓ Batch Normalization 사용
- ✓ 전체학습, epoch 70 이상
- ✓ 훈련 이미지 증식
  - shift / sheering / flip
- ✓ relu / adam 사용



초기 모델 결과

In [7]:
```
# 1번 모델 결과
tt=[]
for i in range(150):
    argmax = np.argmax(output[i])
    tt.append(argmax)
p=list(k.items())
print('1차 결과')
print(tt[0:25], p[0])
print(tt[25:50], p[1])
print(tt[50:75], p[2])
print(tt[75:100], p[3])
print(tt[100:125], p[4])
print(tt[125:150], p[5])
```

```
1차 결과
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0] ('Bus', 0)
[1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1] ('Cargo truck', 1)
[2, 2, 2, 2, 2, 2, 2, 2, 2, 4, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2] ('Motorcycle', 2)
[3, 3, 3, 3, 3, 3, 3, 5, 3, 5, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3] ('Pickup truck', 3)
[5, 5, 5, 5, 5, 3, 5, 5, 5, 2, 5, 4, 4, 5, 5, 5, 4, 4, 5, 4, 5, 5, 4, 4, 5] ('Sedan', 4)
[4, 4, 3, 3, 5, 5, 5, 3, 4, 5, 5, 5, 3, 3, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5] ('Suv', 5)
```

In [9]:
```
# 정확도 계산
accuracy_sum = 0
for i in range(150):
    if tt[i] == tt2[i]:
        accuracy_sum += 1
print(accuracy_sum)
accuracy = (accuracy_sum/150)*100
print(accuracy, '%')
```

```
121
80.66666666666666 %
```

```
-------- Predict -------
{'Bus': 0, 'Cargo truck': 1, 'Motorcycle': 2, 'Pickup truck': 3,
answer :  [1.000 0.000 0.000 0.000 0.000 0.000]
정답 라벨 :  Bus
output :  [0.938 0.007 0.006 0.016 0.031 0.003]
예측 결과 :  Bus
```

Test 정확도 : 80.6%

세단/SUV간의 오차가 높음

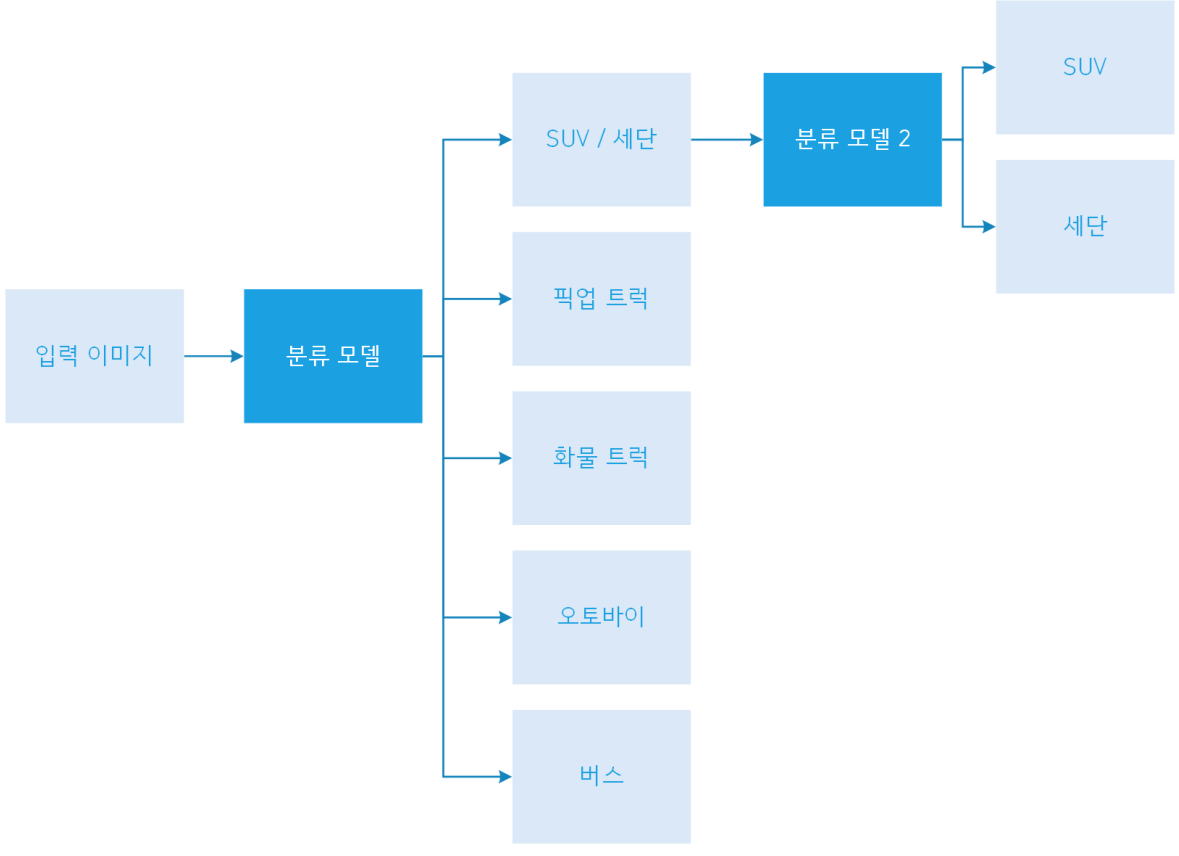
원인 분석

✓ Sedan(Grandeur)

✓ SUV(Veloster)

개선된 모델 구조

- ✓ 분류 모델 추가
  - SUV/세단
  - 동일한 모델
  - class 숫자만 다르게
  - epoch 70 이상



개선된 모델 결과

```
2차 결과
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0] ('Bus', 0)
[1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1] ('Cargo truck', 1)
[2, 2, 2, 2, 2, 2, 2, 2, 2, 4, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2] ('Motorcycle', 2)
[3, 3, 3, 3, 3, 3, 3, 5, 3, 5, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3] ('Pickup truck', 3)
[4, 4, 4, 4, 4, 3, 4, 4, 4, 2, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4] ('Sedan', 4)
[5, 4, 3, 3, 5, 4, 5, 3, 4, 5, 5, 5, 3, 3, 5, 5, 5, 4, 5, 5, 5, 5, 5, 5, 5] ('Suv', 5)
```

```
# 정확도 계산
accuracy_sum = 0
for i in range(150):
    if tt[i] == tt2[i]:
        accuracy_sum += 1
print(accuracy_sum)
accuracy = (accuracy_sum/150)*100
print(accuracy, '%')
```

```
135
90.66666666666666 %
```

```
[14]: print('Network 1')
      print(Label)
      print(output)
      print(" ")

      print('Network 2')
      print(Label2)
      print(output2)
```

```
Network 1
['Bus', 'Cargo truck', 'Motorcycle', 'Pick-up truck', 'Sedan', 'Suv']
[[0.000 0.000 0.000 0.000 0.999 0.000]]

Network 2
['Sedan', 'Suv']
[[0.130 0.870]]
```

✓ 80.6% -> 90.6%

(10% 정확도 향상)

최종 결과

```
-------- Predict -------
OUTPUT :  [0.000 0.000 0.000 0.006 0.007 0.993]
예측 결과 :  Suv
```

- 라벨 없는 이미지들 입력

  결과 라벨을 예측

- 향후 개선 아이디어

  - 사람이 탑승한 오토바이
  - 화물이 없는 화물트럭
  - 더 많은 훈련 셋(현재 80X6개)

Task Sharing

- ***
  - 프로그램 코딩 및 관련 내용 조사
- ***
  - 데이터 수집 및 PPT발표 준비
- ***
  - 아이디어 기획 및 관련내용 조사, 데이터 수집

Q&A

감사합니다

Thank you!