DEVIEW
2019

어디까지 깎아봤니?:
모바일 서비스를 위한 가벼운 이미지 인식/검출용 딥러닝 모델 설계

한동윤
CLOVA AI Research
NAVER

0. Introduction

DEVIEW 2019

한동윤, CLOVA AI Research

Current research areas:
- Object classification & detection 기술 연구
- Scene text detection and recognition (OCR) 기술 연구

Homepage: http://dongyoonhan.github.io
Google scholar: https://scholar.google.com/citations?user=jcP7m1QAAAAJ&hl=en

0. Introduction – 이 talk에서 할 이야기

DEVIEW 2019

새로 만든 성능 좋은 가벼운 딥러닝 모델에 대한 소개와 관련 이야기를 하려 합니다

세부 내용으로는,

- 큰 데이터셋을 통한 딥러닝 모델 개발 전반에 관한 이야기
- 서비스에 딥러닝 모델을 적용할때 강력한 모델의 필요성?
- 강력한 모델을 왜 연구해야하는 것인가?
- 새로 개발한 강력한 light-weight (가벼운) 딥 러닝 모델 (+backbone)에 대한 이야기
- 각종 팁들 나눔

CONTENTS

DEVIEW 2019

1. Deep Neural Network의 출현으로 도래한 인공지능 (AI) 시대
2. 딥러닝 모델 (backbone)을 통한 이미지 분류/검출기 트레이닝의 개괄
3. 최신 가벼운 딥러닝 모델들 소개와 몇가지 이슈
4. 새로 개발한 가벼운 모델과 성능 소개
5. 신 모델을 활용한 finetuning 성능과 실 서비스 적용 예 + 추가 팁

DEVIEW
2019

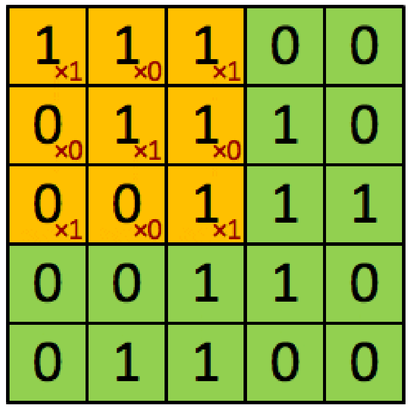
1. Deep Neural Network의 출현으로 도래한 인공지능 (AI) 시대

1.0 Convolutional Neural Network (CNN)

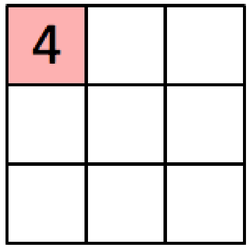
DEVIEW 2019

- CNN: **Y. LeCun**에 의해 처음 개발 (LeNet, 1989년도), Hidden layer로 convolution layer을 사용
- Convolution layer의 개수가 많은 모델: Deep Convolutional Neural Network (Deep CNN)

  예) VGG16 [1]:

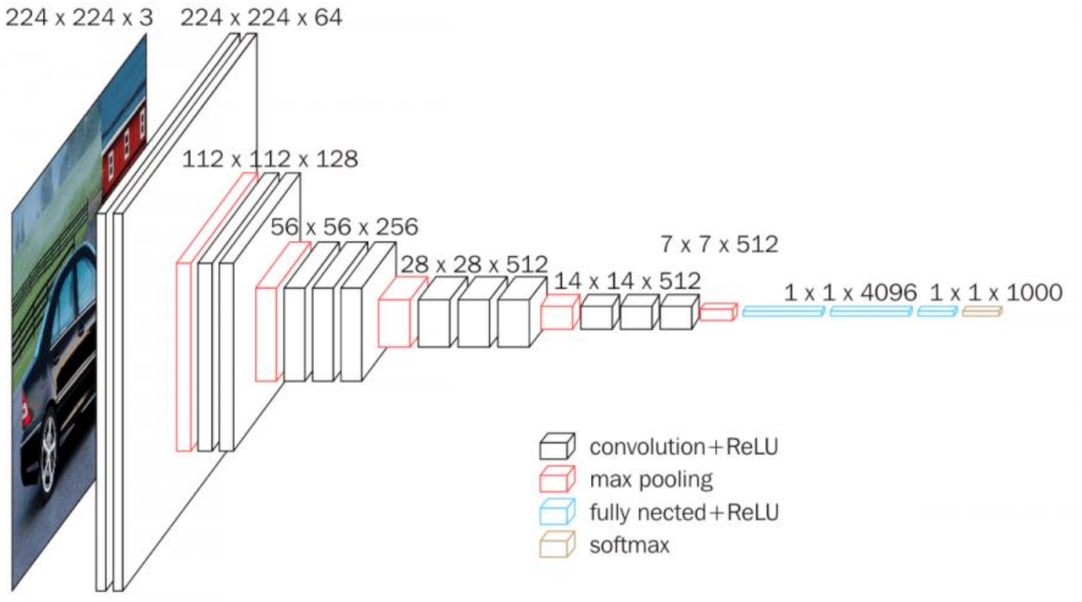


Image



Convolved Feature

이미지 출처: https://giphy.com/gifs/blog-daniel-keypoints-i4NjAwytgIRDW



이미지 출처: https://neurohive.io/en/popular-networks/vgg16/

[1] K. Simonyan and A. Zisserman. “Very deep convolutional networks for large-scale image recognition”. In ICLR, 2015

1.1 AlexNet의 등장

DEVIEW 2019

- Neural Network 빙하기를 녹이는 강력한 한방:
  - Alex Khrizhebsky가 최초로 GPU 를 사용하여 Deep CNN 트레이닝 (학습)
- **NIPS 2012** 에 논문으로 제출, 개발된 모델이 그 유명한 **AlexNet** (아래는 논문)

ImageNet Classification with Deep Convolutional Neural Networks

Alex Krizhevsky
University of Toronto
kriz@cs.utoronto.ca

Ilya Sutskever
University of Toronto
ilya@cs.utoronto.ca

Geoffrey E. Hinton
University of Toronto
hinton@cs.utoronto.ca

Abstract

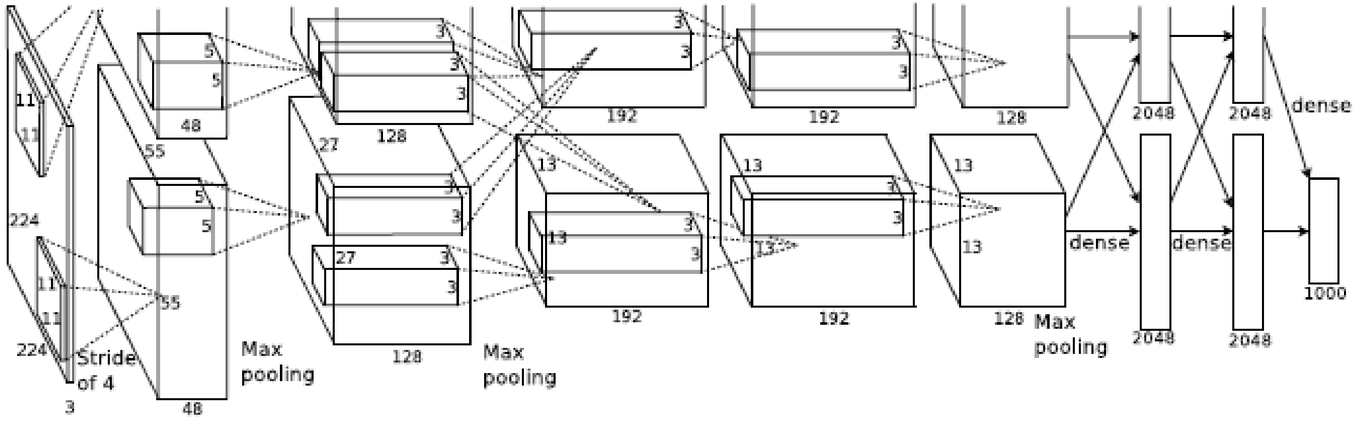
We trained a large, deep convolutional neural network to classify the 1.2 million high-resolution images in the ImageNet LSVRC-2010 contest into the 1000 different classes. On the test data, we achieved top-1 and top-5 error rates of 37.5% and 17.0% which is considerably better than the previous state-of-the-art. The neural network, which has 60 million parameters and 650,000 neurons, consists of five convolutional layers, some of which are followed by max-pooling layers, and three fully-connected layers with a final 1000-way softmax. To make training faster, we used non-saturating neurons and a very efficient GPU implementation of the convolution operation. To reduce overfitting in the fully-connected layers we employed a recently-developed regularization method called "dropout"

1.1 AlexNet의 등장

DEVIEW 2019

- 8개의 weight layer의 Deep CNN, 추가적으로 ReLU, dropout 개발, 적용 [1]
- ImageNet 데이터셋에서 분류 성능 대폭 향상 (top-5 acc.기준 +10% 이상)

→ 10년 이상 지속되었던 성능 정체를 단번에 해소!



Model	Top-1 (val)	Top-5 (val)	Top-5 (test)
SIFT + FVs [7]	—	—	*26.2%*
1 CNN	40.7%	18.2%	—
5 CNNs	38.1%	16.4%	**16.4%**
1 CNN*	39.0%	16.6%	—
7 CNNs*	36.7%	15.4%	**15.3%**

Table 2: Comparison of error rates on ILSVRC-2012 validation and test sets. In *italics* are best results achieved by others. Models with an asterisk* were "pre-trained" to classify the entire ImageNet 2011 Fall release. See Section 6 for details.

[1] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks",

1.2 도래한 AI 시대

DEVIEW 2019

Object detection in the Wild [1]

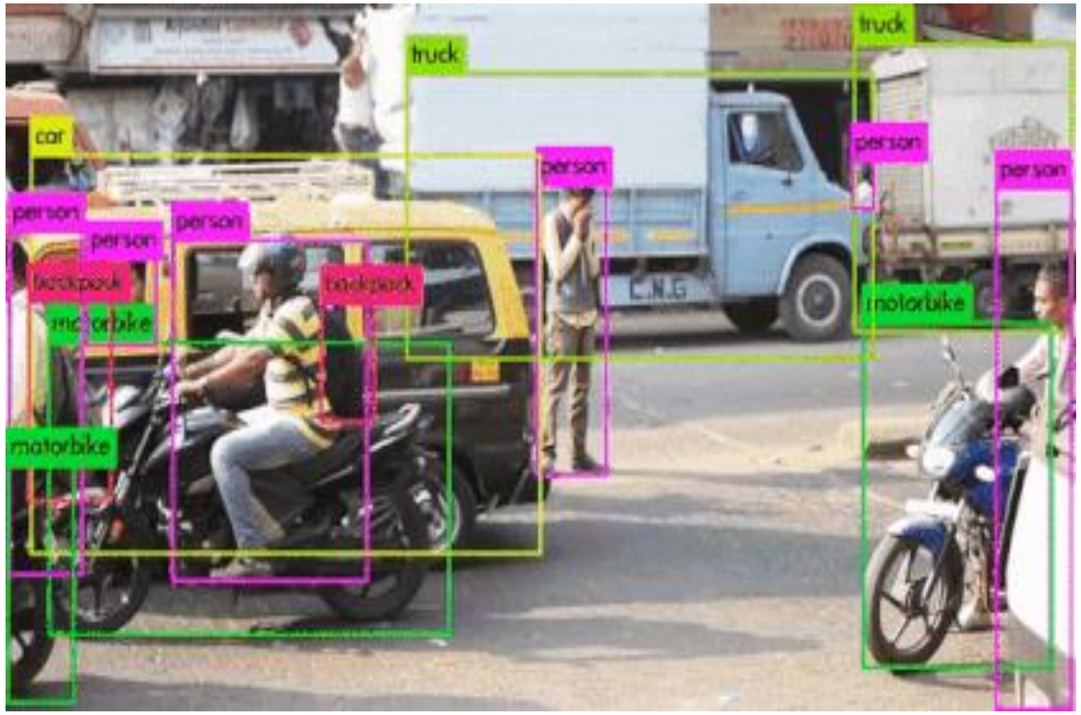
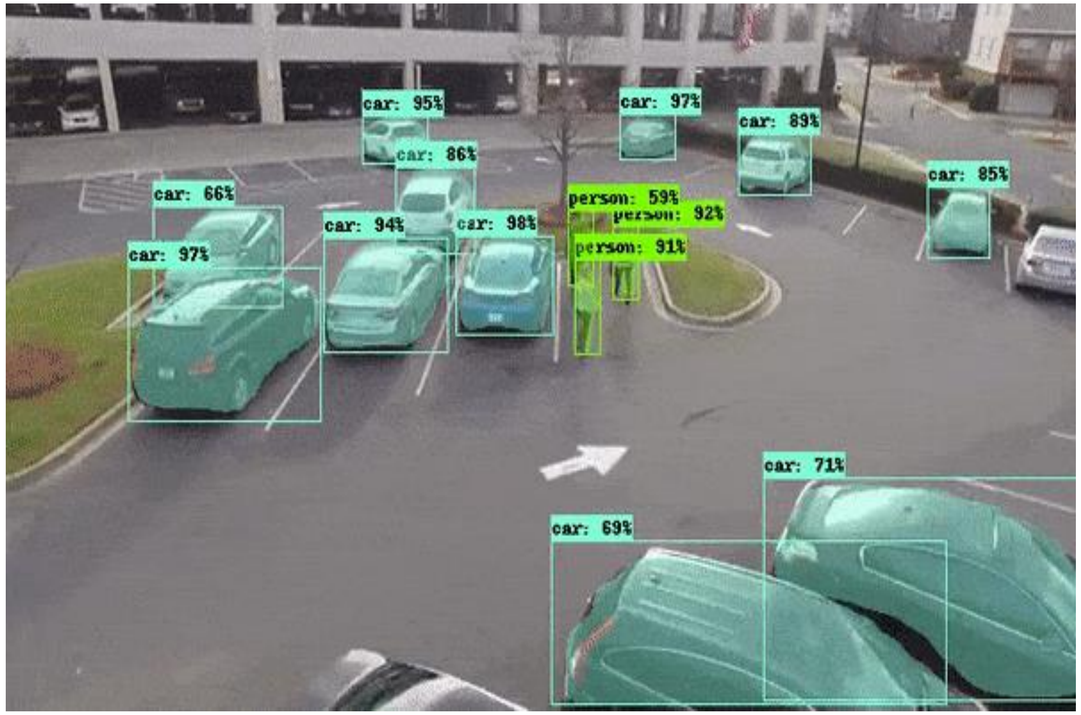


Image Segmentation in the Wild [2]



이미지 출처: [1] https://towardsdatascience.com/using-tensorflow-object-detection-to-do-pixel-wise-classification-702bf2605182

이미지 출처: [2] https://towardsdatascience.com/review-yolov3-you-only-look-once-object-detection-eab75d7a1ba6

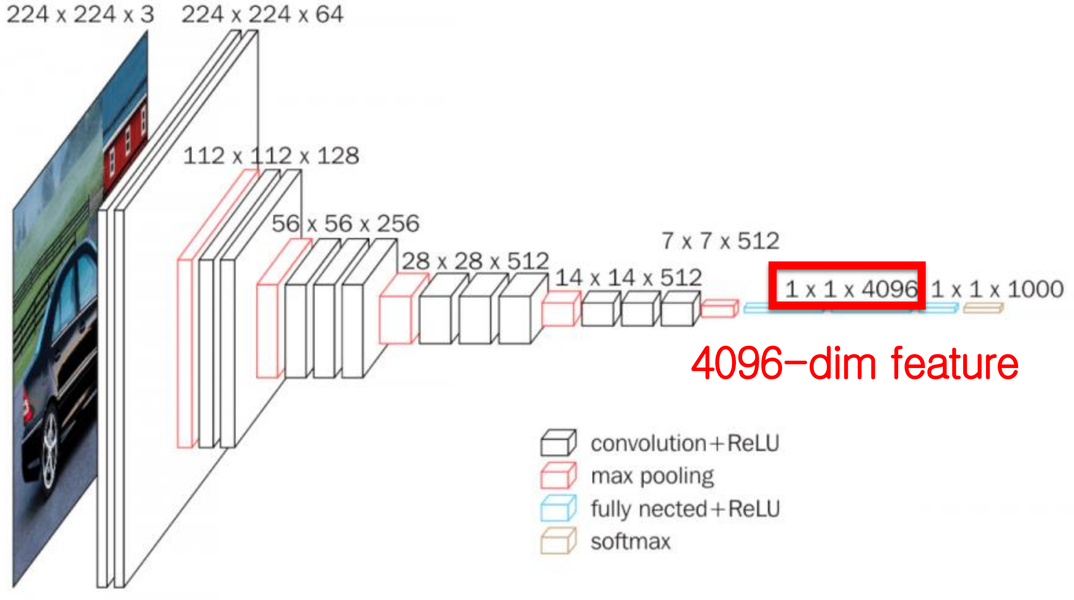
DEVIEW
2019

2. 딥러닝 모델 (backbone)을 통한 이미지 분류/검출기 트레이닝의 개괄

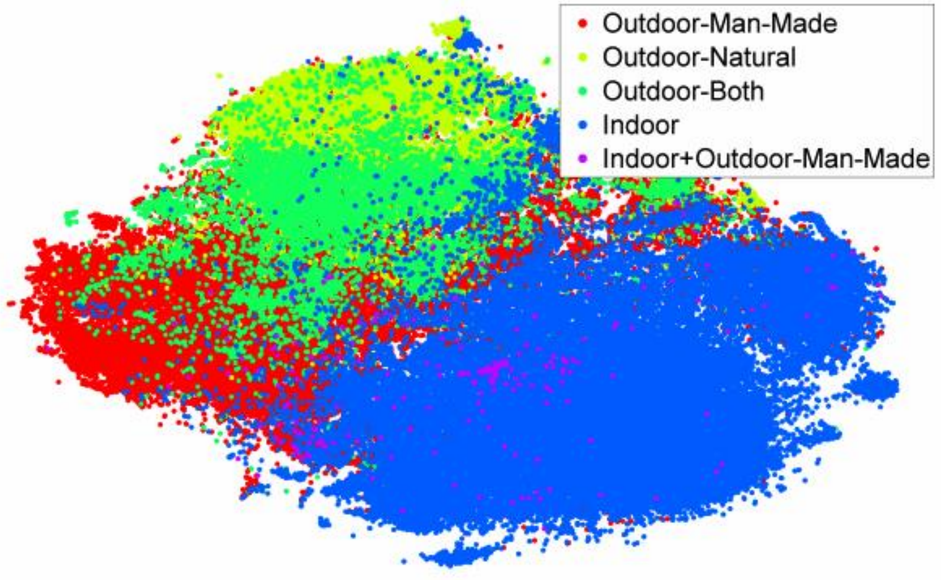
2.0 Backbone이란 = feature extractor

DEVIEW 2019

- Backbone (모델) 이란? 큰 데이터셋 (ImageNet 등)에 기 학습된 (pretrained) 딥러닝 모델을 칭함:
- Backbone 은 feature extractor (특징 추출기) 입니다:
  - Backbone에 이미지를 넣으면 feature가 추출: 아래 그림은 추출된 feature (특징)을 plot 한 것



이미지 출처: https://neurohive.io/en/popular-networks/vgg16/



이미지 출처: https://arxiv.org/pdf/1310.1531.pdf

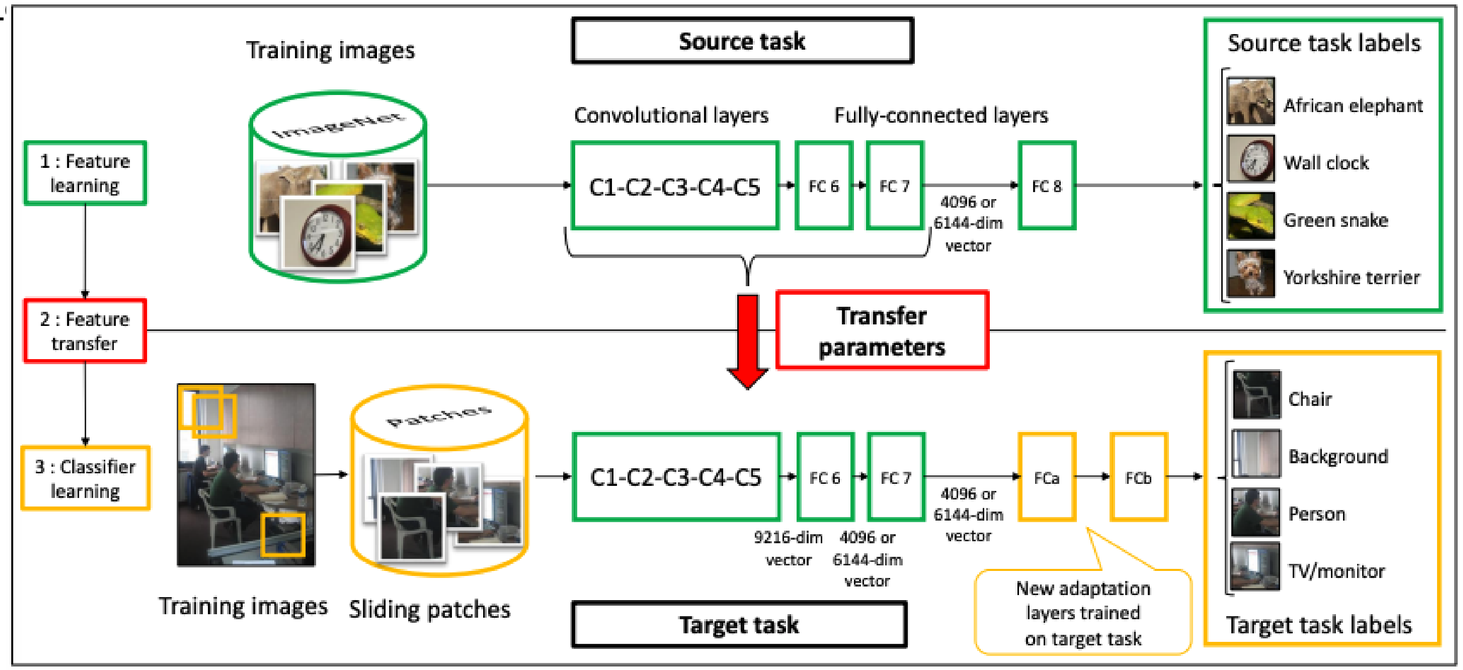
※ 용어의 변화: 뉴럴 네트워크 → 딥 뉴럴 네트워크 → 딥 네트워크 → 딥 러닝 모델 → 딥 모델 → 모델

2.0 Transfer learning이란?

DEVIEW 2019

- Backbone의 feature을 시작점으로 target task에서 트레이닝을 시작하는 기법
  - Backbone의 weight 파라미터들을 그대로 전이 시켜 (transfer) target task 데이터에서 학습 시작
- 이를 수행한 첫 연구:

Oquab et al., - L[illegible] Neural Networks, CVPR 2014

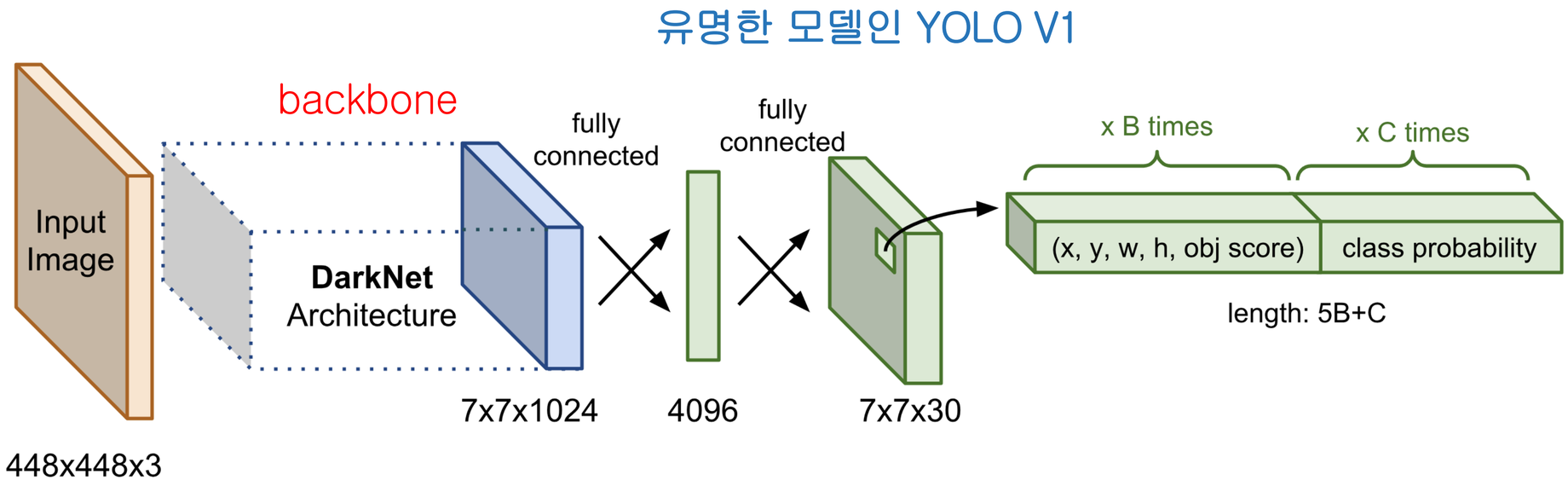


이 미 지 출 처 : https://www.cv-foundation.org/openaccess/content_cvpr_2014/papers/Oquab_Learning_and_Transferring_2014_CVPR_paper.pdf

2.0 Finetuning이란?

DEVIEW 2019

- 아래와 같은 object detector (YOLO V1)을 transfer learning을 통해 트레이닝할 때, 추가된 layer을 학습하기 위해 backbone의 weight를 고정하고 미세하게 튜닝하는 것을 finetuning이라고 합니다



이미지 출처: https://lilianweng.github.io/lil-log/2018/12/27/object-detection-part-4.html

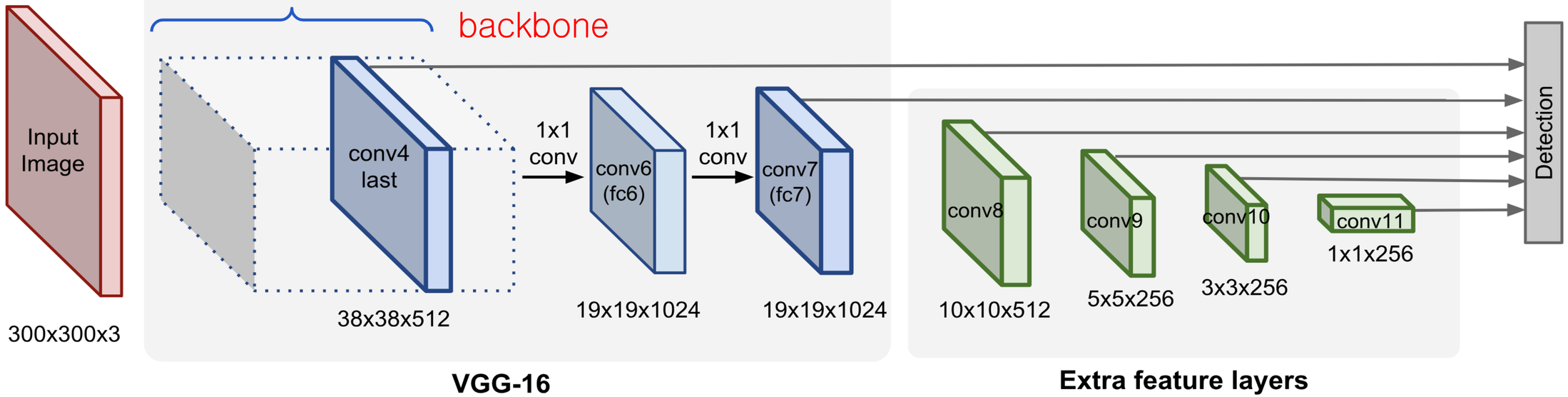
※ 용어의 변화: 뉴럴 네트워크 → 딥 뉴럴 네트워크 → 딥 네트워크 → 딥 러닝 모델 → 딥 모델 → 모델

2.0 Finetuning이란?

DEVIEW 2019

- 아래와 같은 object detector (SSD)을 transfer learning을 통해 트레이닝할 때, 추가된 layer을 학습하기 위해 backbone의 weight를 고정하고 미세하게 튜닝하는 것을 finetuning이라고 합니다
- backbone의 weight 파라미터를 고정하지 않고 트레이닝 하는 경우도 많아, 그 경우도 fir

유명한 모델인 Single Shot Detector



이미지 출처: https://lilianweng.github.io/lil-log/2018/12/27/object-detection-part-4.html

※ 용어의 변화: 뉴럴 네트워크 → 딥 뉴럴 네트워크 → 딥 네트워크 → 딥 러닝 모델 → 딥 모델 → 모델

2.1 서비스에 딥러닝 모델을 적용한다면

DEVIEW 2019

- 실 서비스에 딥러닝 모델을 적용하려면 (예: detection 서비스를 딥러닝 모델로 서빙 한다면)
  - Pretrained backbone을 기반 (즉 큰 데이터를 가지고 미리 backbone을 학습)으로 finetuning을 활용해서 target task의 데이터셋에서 학습하는 것이 성능을 확보하는 가장 좋은 방법 입니다

- 그런데 이러한 target task에서의 성능을 확보하는데 중요한 이슈는 크게,
  - 어떤 큰 데이터를 이용해 딥러닝 모델을 pretraining 할지 여부 – 데이터셋 이슈
  - 어떤 딥러닝 모델을 써서 어떻게 트레이닝 할지 – 딥러닝 모델 선택 & 학습 이슈
  - Pretrained backbone 모델의 성능이 얼마나 뛰어난지 - backbone의 성능 이슈
  - 각 과정에서 트레이닝 세팅 정밀화 여부 - 트레이닝 세팅 이슈

- 위 과정들을 뒤이어 설명 드리겠습니다 (target task로 **object detection**을 예로 들겠습니다)

2.1 딥 러닝 물체 분류/검출기 트레이닝 개요도

DEVIEW 2019

1. 큰 데이터셋 확보

2. 딥 러닝 모델 선택, 큰 데이터에 학습 (pretrain)

3. Pretrained backbone 의 성능 관찰 (evaluation)

4. 목표 데이터셋의 학습 데이터로 학습 (finetuning)

5. 목표 데이터셋에서 검출기 성능 관찰 (evaluation)

2.2 큰 데이터 셋?

DEVIEW 2019

1. 큰 데이터셋 확보

ImageNet-1k (or 5k), OpenImage

2. 딥 러닝 모델 선택, 큰 데이터에 학습 (pretrain)

3. Pretrained backbone 의 성능 관찰 (evaluation)

4. 목표 데이터셋의 학습 데이터로 학습 (finetuning)

5. 목표 데이터셋에서 검출기 성능 관찰 (evaluation)

2.2 큰 데이터 셋?: ImageNet

DEVIEW 2019

- **ImageNet-1k** (일반적으로 불리는 ImageNet, http://www.image-net.org):
  - 원래의 데이터셋인 ImageNet-22k (데이터 개수: 14,197,122) 의 subset 입니다.
  - 데이터셋 구성:
    - Training set: ImageNet-1k (class 개수: 1k개, 1,281,167개~=1M개의 트레이닝 이미지)
    - Validation set: ImageNet-1k (50,000개 valdiation 데이터)
    - Test set: 매년 ImageNet 대회 - **ImageNet Large Scale Visual Recognition Challenge (ILSVRC)** 에서 공개

이미지 출처: http://www.image-net.org

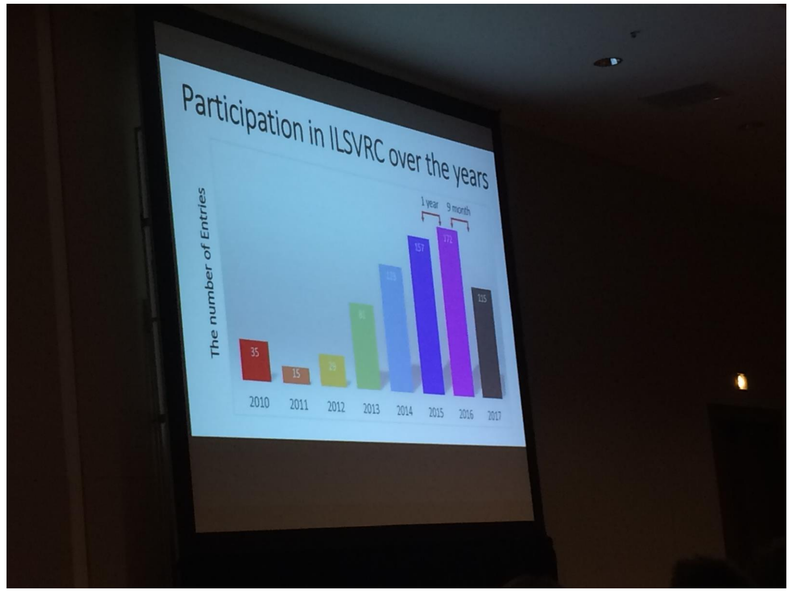
2.2 큰 데이터 셋?: ImageNet

DEVIEW 2019

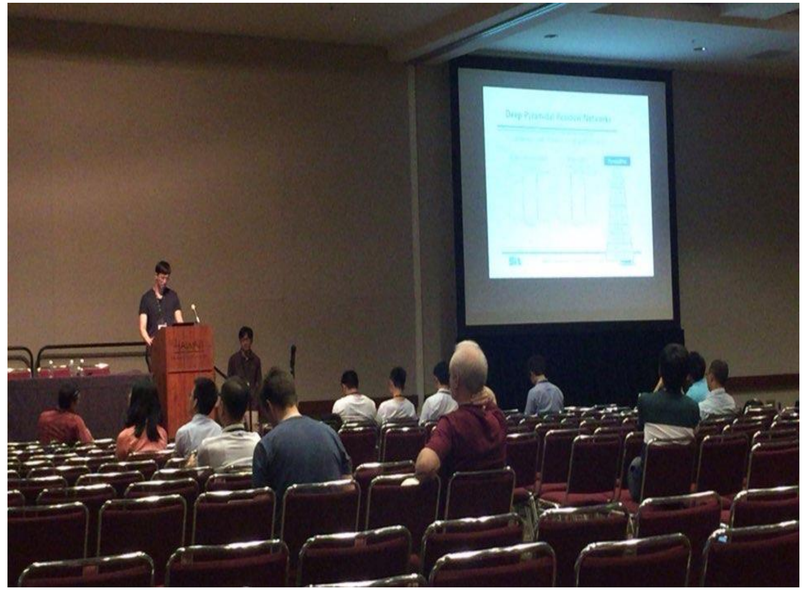
IMAGENET **Large Scale Visual Recognition Challenge 2017 (ILSVRC2017)**

- ILSVRC: 매년 ImageNet-1k 데이터를 통해 물체 인식/검출 등을 겨루는 challenge
- 마지막 ImageNet 대회 (**ILSVRC 2017**)의 전경:

Fei-Fei Li 교수의 강연



대회의 최종 마무리 중



한동윤님의 결과 발표

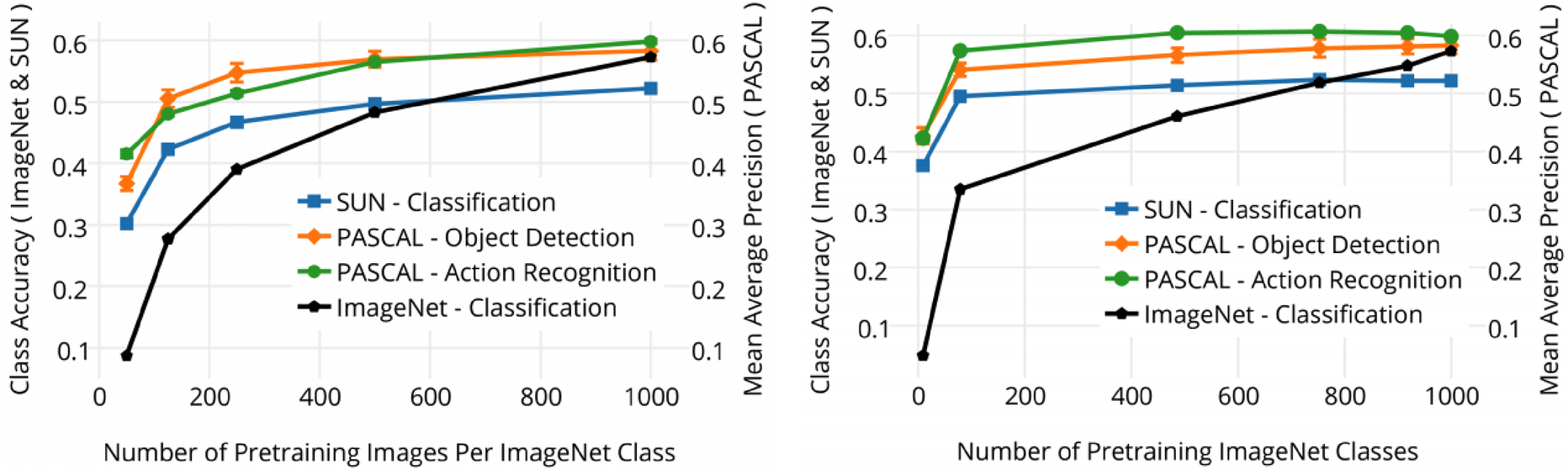
2.2 큰 데이터 셋?: ImageNet

- ImageNet-1k외에도 ImageNet-5k 데이터셋이 있습니다
- OpenImage 데이터셋 (V5버전이 나오고 최근에 뜨기 시작한 데이터셋)
- Instagram 데이터셋 (ECCV 2018에 등장했지만, 여전히 미공개 데이터셋)
- 특히 이중에서도 ImageNet-1k가 가장 보편적, 일반적으로 널리 사용됩니다

- **이유는 경험적으로 잘된다는 것이 알려져 있고, 그 외에 연구 결과들이 있습니다**

2.2 큰 데이터 셋?: ImageNet

- **뒷받침 논문 1:**
  → Huh et al., What makes ImageNet good for transfer learning?, arxiv 2017
  → ImageNet에서 1) class별로 많은 이미지를 쓸수록, 2) 많은 class 를 써서 트레이닝 할수록 모델이 transfer learning이 잘된다는 결과:



이미지 출처: https://arxiv.org/pdf/1608.08614.pdf

2.2 큰 데이터 셋?: ImageNet

- **뒷받침 논문 2:**

→ Recht et al., Do ImageNet Classifiers Generalize to ImageNet?, ICML 2019

→ ImageNet에서 좋은 성능 (Orig. Accuracy)을 보이는 모델은 본 논문에서 만든 ImageNet과 흡사한

새 데

ImageNet Top-1

Orig. Rank	Model	Orig. Accuracy	New Accuracy	Gap	New Rank	Δ Rank
1	pnasnet_large_tf	82.9 [82.5, 83.2]	72.2 [71.3, 73.1]	10.7	3	-2
4	nasnetalarge	82.5 [82.2, 82.8]	72.2 [71.3, 73.1]	10.3	1	3
21	resnet152	78.3 [77.9, 78.7]	67.0 [66.1, 67.9]	11.3	21	0
23	inception_v3_tf	78.0 [77.6, 78.3]	66.1 [65.1, 67.0]	11.9	24	-1
30	densenet161	77.1 [76.8, 77.5]	65.3 [64.4, 66.2]	11.8	30	0
43	vgg19_bn	74.2 [73.8, 74.6]	61.9 [60.9, 62.8]	12.3	44	-1
64	alexnet	56.5 [56.1, 57.0]	44.0 [43.0, 45.0]	12.5	64	0
65	fv_64k	35.1 [34.7, 35.5]	24.1 [23.2, 24.9]	11.0	65	0

표 출처: https://arxiv.org/pdf/1902.10811.pdf

2.3 모델 선택

DEVIEW 2019

1. 큰 데이터셋 확보

2. **딥 러닝 모델 선택**, 큰 데이터에 **학습 (pretrain)** ◀ ResNet-50, MobileNetV1, …

3. Pretrained backbone 의 성능 관찰 (evaluation)

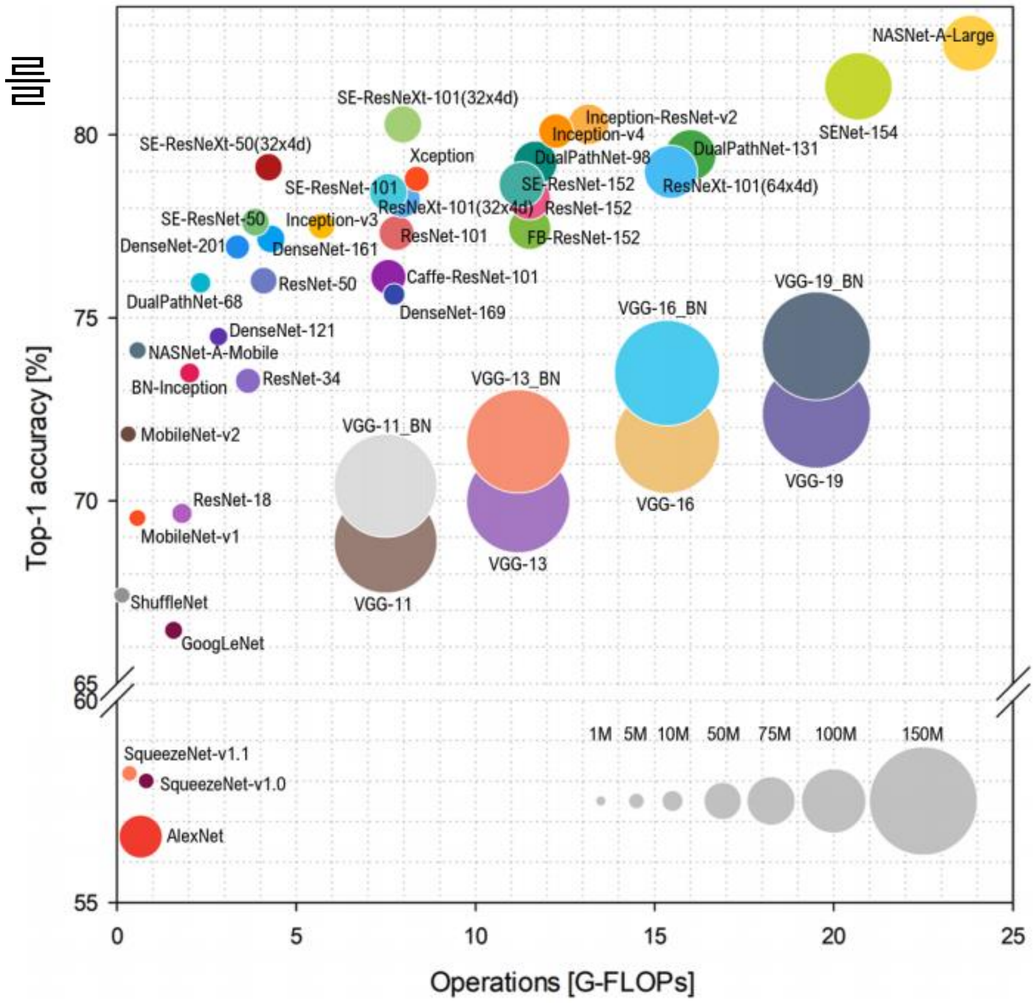
4. 목표 데이터셋의 학습 데이터로 학습 (finetuning)

5. 목표 데이터셋에서 검출기 성능 관찰 (evaluation)

2.3 모델 선택 & 학습

- 어떤 딥러닝 모델을 선택해서 큰 데이터 셋에서 학습할지를 정해야 합니다
- ImageNet이 큰 데이터 셋으로 선택되었다면 유명 pretrained 모델 (ImageNet에 학습된 모델) 이 많이 공개되어 있으므로 선택하면 됩니다

※ 직접 ImageNet 정도 크기의 데이터셋에 트레이닝 하려면
"일반 세팅"으로 4/8GPU 로 트레이닝 시에
최소 3~7일 시간이 소요됩니다 (GPU 성능에 의존)

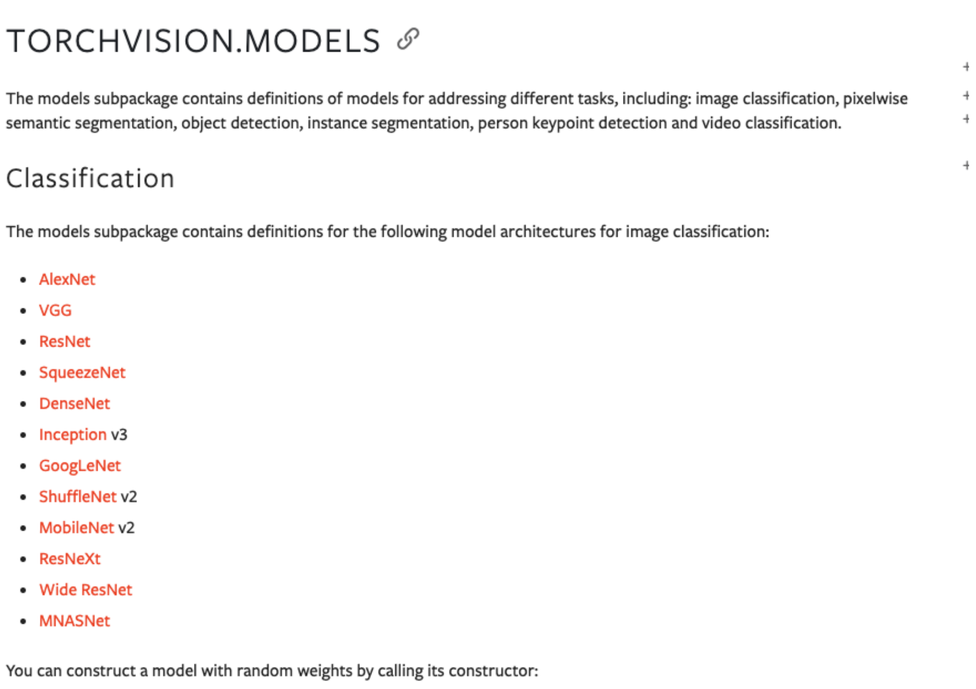


유명 모델들의 ImageNet 성능 비교 그래프

이미지 출처: https://arxiv.org/pdf/1810.00736.pdf

2.3 모델 선택 & 학습: model-zoo

- 공개된 model-zoo (모델들의 동물원 맞습니다):
  - 빠른 deploy를 위해 TensorFlow, PyTorch, Mxnet등에서 model을 모아놓은 페이지를 공개:



TORCHVISION.MODELS

The models subpackage contains definitions of models for addressing different tasks, including: image classification, pixelwise semantic segmentation, object detection, instance segmentation, person keypoint detection and video classification.

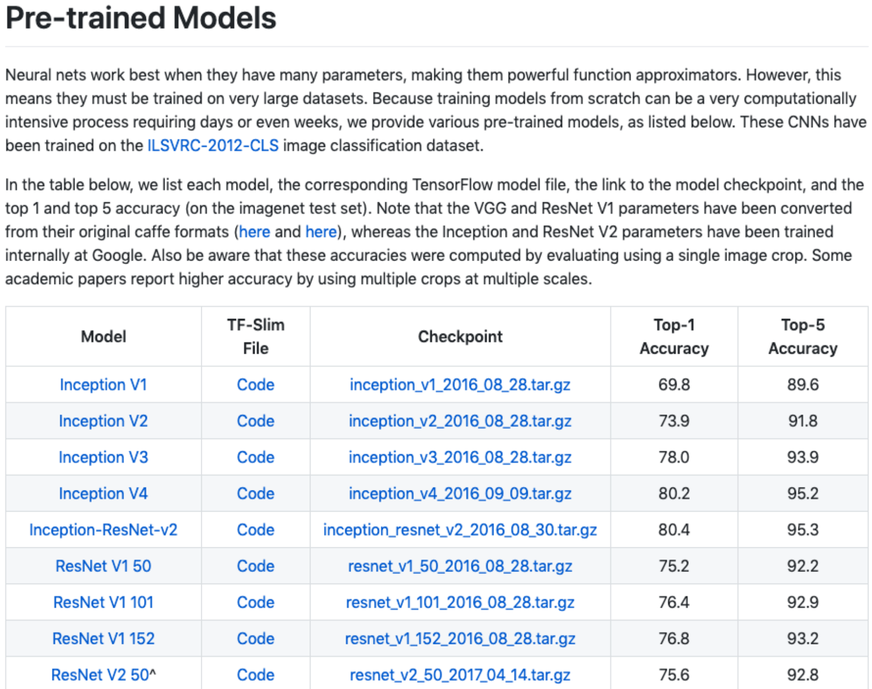
Classification

The models subpackage contains definitions for the following model architectures for image classification:

- AlexNet
- VGG
- ResNet
- SqueezeNet
- DenseNet
- Inception v3
- GoogLeNet
- ShuffleNet v2
- MobileNet v2
- ResNeXt
- Wide ResNet
- MNASNet

You can construct a model with random weights by calling its constructor:

PyTorch의 model-zoo



Pre-trained Models

Neural nets work best when they have many parameters, making them powerful function approximators. However, this means they must be trained on very large datasets. Because training models from scratch can be a very computationally intensive process requiring days or even weeks, we provide various pre-trained models, as listed below. These CNNs have been trained on the ILSVRC-2012-CLS image classification dataset.

In the table below, we list each model, the corresponding TensorFlow model file, the link to the model checkpoint, and the top 1 and top 5 accuracy (on the imagenet test set). Note that the VGG and ResNet V1 parameters have been converted from their original caffe formats (here and here), whereas the Inception and ResNet V2 parameters have been trained internally at Google. Also be aware that these accuracies were computed by evaluating using a single image crop. Some academic papers report higher accuracy by using multiple crops at multiple scales.

Model	TF-Slim File	Checkpoint	Top-1 Accuracy	Top-5 Accuracy
Inception V1	Code	inception_v1_2016_08_28.tar.gz	69.8	89.6
Inception V2	Code	inception_v2_2016_08_28.tar.gz	73.9	91.8
Inception V3	Code	inception_v3_2016_08_28.tar.gz	78.0	93.9
Inception V4	Code	inception_v4_2016_09_09.tar.gz	80.2	95.2
Inception-ResNet-v2	Code	inception_resnet_v2_2016_08_30.tar.gz	80.4	95.3
ResNet V1 50	Code	resnet_v1_50_2016_08_28.tar.gz	75.2	92.2
ResNet V1 101	Code	resnet_v1_101_2016_08_28.tar.gz	76.4	92.9
ResNet V1 152	Code	resnet_v1_152_2016_08_28.tar.gz	76.8	93.2
ResNet V2 50^	Code	resnet_v2_50_2017_04_14.tar.gz	75.6	92.8

Tensorflow의 model-zoo

2.3 모델 선택 & 학습: 기존 모델 사용 시

- **ImageNet 데이터가 아닌 다른 큰 데이터에 딥러닝 모델을 학습하려 할 때:**
  - 이때에도 ImageNet 성능 기준으로 좋은 딥러닝 모델을 선택해서 사용하는 것이 좋습니다

  - 즉 ResNet50, VGG16등의 모델을 선택한 이후 사용하실 큰 데이터에 트레이닝 하는 것이 좋습니다

  - 당연히 상황에 따라 모델에 적절한 변화를 가할 수 있습니다 (변화를 시키는 과정이 새로운 모델 개발 과정)

  - “상황에 따라” 추천할만한 모델은 appendix에 있으니 확인하시면 되겠습니다

2.3 모델 선택 & 학습: 새로운 모델 개발 시

DEVIEW 2019

- **이미 개발된 딥 러닝 구조를 쓰지 않고 딥러닝 모델을 새롭게 개발한다면**
  - 1) 사용할 큰 데이터 셋에 맞는 좋은 모델 (성능/속도 측면) 을 먼저 설계/연구 (반복 실험 필요)
  - 2) 그리고 학습 방법에 따라 성능이 크게 차이 (정확도 기준으로 3~5% 이상) 나게 되기 때문에 학습 방법을 정밀하게 최적화 시도
  - 3) 최적화 후 성능을 기존 모델과 비교 후 좋은 성능이 나오는지 확인 (아니라면 1)로 돌아갑니다)

- 본 talk에서 제시할 새로운 모델은 이와 같은 노력 (깎는 노력) 이 들어간 모델입니다:
  → 즉 모델을 깎는다? 성능 좋은 모델 설계/개발 + 트레이닝 스킴 최적화

2.4 Backbone의 성능 이슈

1. 큰 데이터셋 확보

2. 딥 러닝 모델 선택, 큰 데이터에 학습 (pretrain)

3. Pretrained Backbone 의 성능 관찰 (evaluation)

4. 목표 데이터셋의 학습 데이터로 학습 (finetuning)

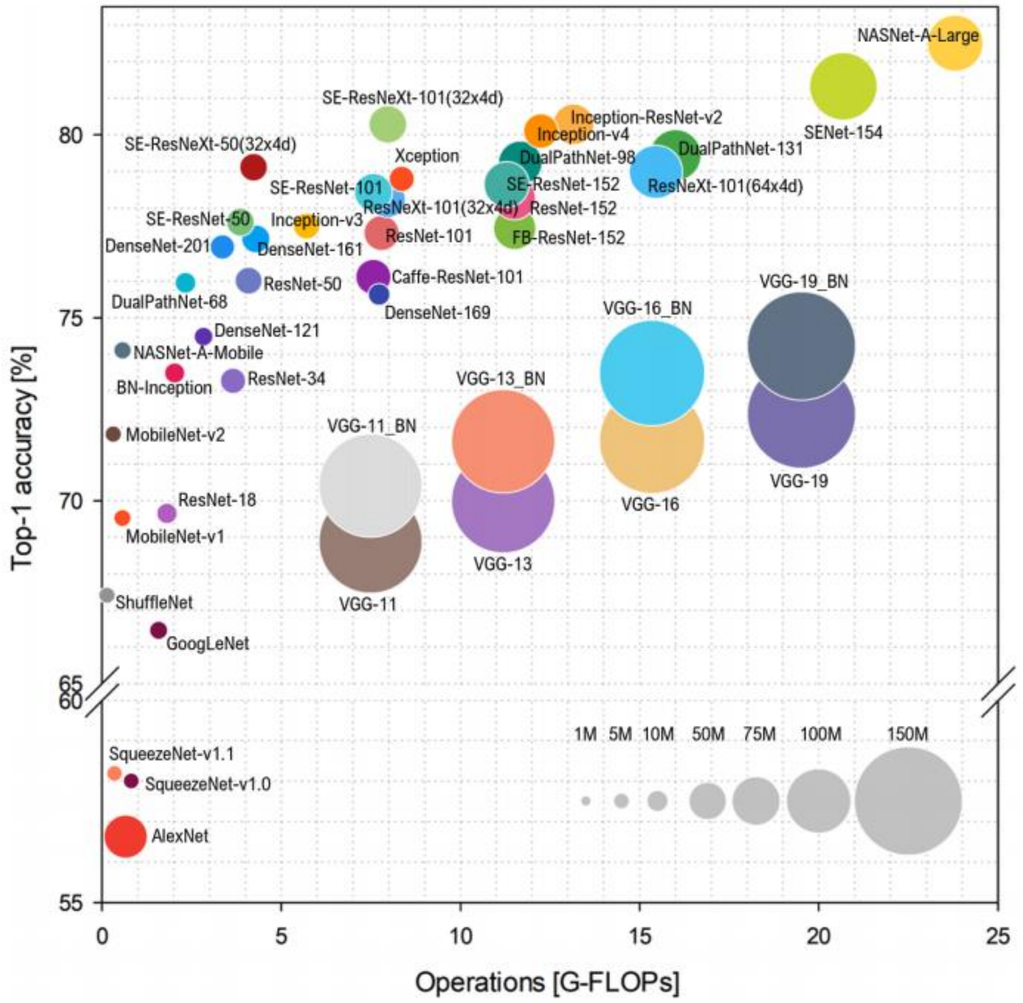
5. 목표 데이터셋에서 검출기 성능 관찰 (evaluation)

2.4 Backbone의 성능

DEVIEW 2019

- **Pretrained backbone의 성능**= ImageNet 데이터셋 분류 성능 (**오른쪽 그래프** 참조)

Q) 강력한 backbone을 사용해야 하는 구체적 이유?



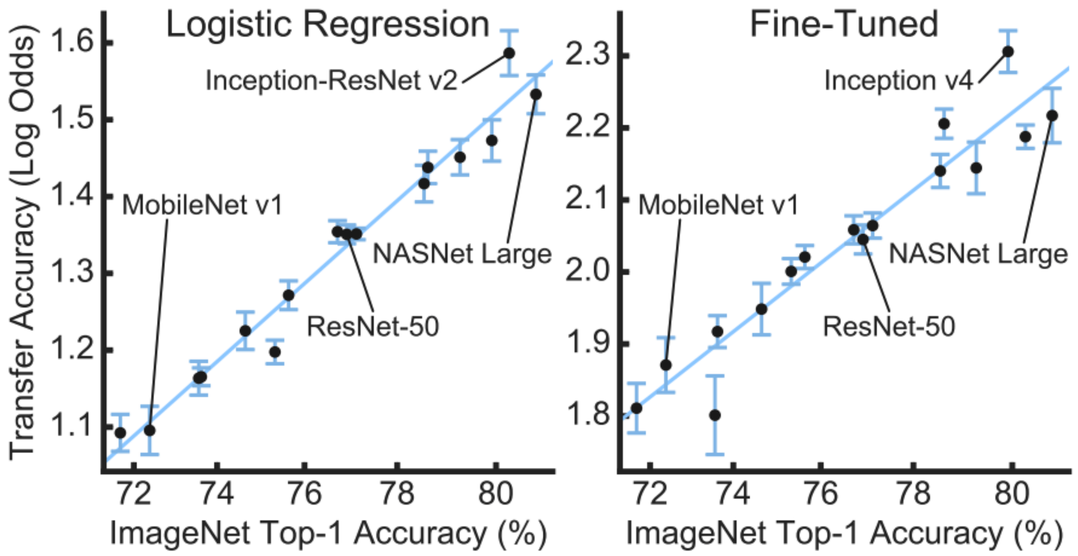
유명 모델들의 ImageNet 성능 비교 그래프

이미지 출처: https://arxiv.org/pdf/1810.00736.pdf

2.4 강력한 성능의 Backbone에 관하여

- **Q) 강력한 backbone을 사용 해야하는 구체적 이유?**

- → Kornblith et al., Do Better ImageNet Models Transfer Better?, CVPR 2019



→ Backbone의 성능과 Target task의 성능은 비례

이미지 출처: https://arxiv.org/abs/1805.08974

2.4 강력한 성능의 Backbone에 관하여

- **Q) 강력한 backbone을 사용 해야하는 구체적 이유?**
- Object detection을 예로 들면:
  - 아래는 Faster RCNN, Libra RCNN 결과 (Libra R-CNN (CVPR 2019) 논문에서 발췌):

Faster R-CNN*	ResNet-50-FPN	1×	36.2	58.5	38.9	21.0	38.9	45.3
Faster R-CNN*	ResNet-101-FPN	1×	38.8	60.9	42.1	22.6	42.4	48.5
Faster R-CNN*	ResNet-101-FPN	2×	39.7	61.3	43.4	22.1	43.1	50.3
Faster R-CNN*	ResNeXt-101-FPN	1×	41.9	63.9	45.9	25.0	45.3	52.3
RetinaNet*	ResNet-50-FPN	1×	35.8	55.3	38.6	20.0	39.0	45.1
Libra R-CNN (ours)	ResNet-50-FPN	1×	**38.7**	59.9	42.0	22.5	41.1	48.7
Libra R-CNN (ours)	ResNet-101-FPN	1×	**40.3**	61.3	43.9	22.9	43.1	51.0
Libra R-CNN (ours)	ResNet-101-FPN	2×	**41.1**	62.1	44.7	23.4	43.7	52.5
Libra R-CNN (ours)	ResNeXt-101-FPN	1×	**43.0**	64.0	47.0	25.3	45.6	54.6
Libra RetinaNet (ours)	ResNet-50-FPN	1×	**37.8**	56.9	40.5	21.2	40.9	47.7

→ ResNet-50-FPN<ResNet-101-FPN<ResNeXt-101-FPN 순으로 backbone 성능이 좋음

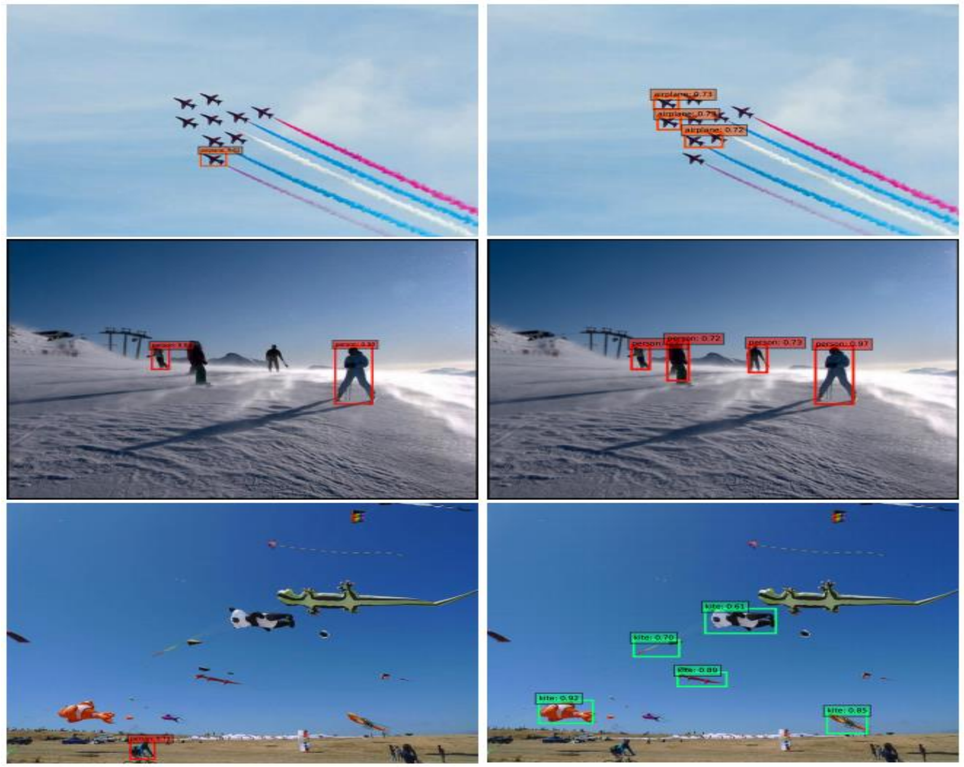
→ 모델별 ImageNet 성능차이가 1~2%정도 나면 target task 성능차이도 2~3%이상

2.4 강력한 성능의 Backbone에 관하여

DEVIEW 2019

- **Q) 강력한 backbone을 사용 해야하는 구체적 이유?**

왼쪽 오른쪽



이미지 출처: https://arxiv.org/pdf/1701.06659.pdf

* 왼쪽, 오른쪽 column – 성능이 다른 두 모델의 visualization 결과

→ 모델 간 ImageNet classification 성능 차이가 1~2% 정도가 있고, COCO2017 detection 성능 차이도 2~3%

→ 그러나 실제 detection 결과는 매우 차이가 큽니다!

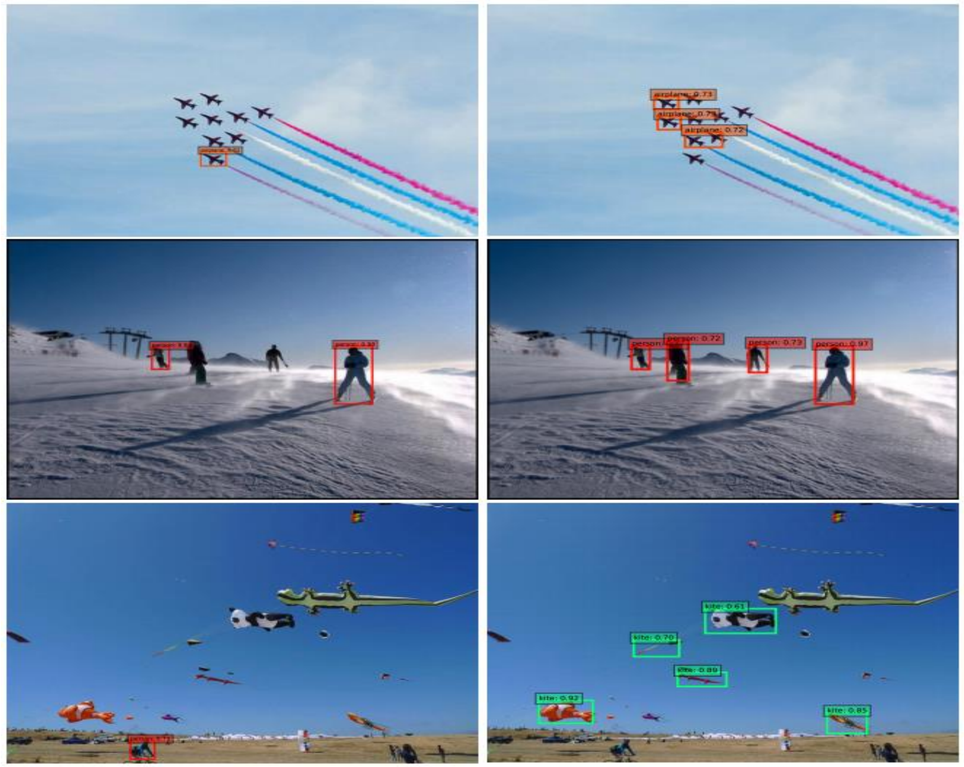
(비행기, 사람, 연을 더 잘 찾기 시작함)

2.4 강력한 성능의 Backbone에 관하여

DEVIEW 2019

- **Q) 강력한 backbone을 ~~사용~~ 개발 해야하는 구체적 이유?**

왼쪽 오른쪽



이미지 출처: https://arxiv.org/pdf/1701.06659.pdf

* 왼쪽, 오른쪽 column – 성능이 다른 두 모델의 visualization 결과

→ 모델 간 ImageNet classification 성능 차이가 1~2% 정도가 있고, COCO2017 detection 성능 차이도 2~3%

→ 그러나 실제 detection 결과는 매우 차이가 큽니다!
(비행기, 사람, 연을 더 잘 찾기 시작함)

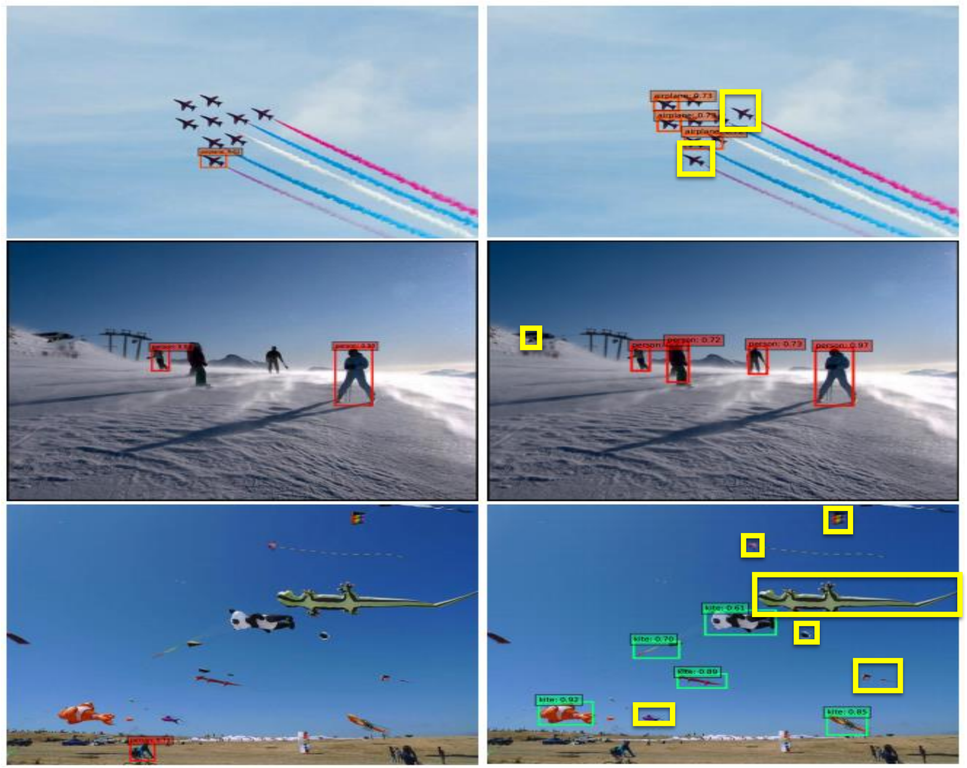
→ 만약 더 좋은 성능의 backbone 모델을 개발한다면?

2.4 강력한 성능의 Backbone에 관하여

DEVIEW 2019

- **Q) 강력한 backbone을 개발 해야하는 구체적 이유?**

왼쪽 오른쪽



이미지 출처: https://arxiv.org/pdf/1701.06659.pdf

* 왼쪽, 오른쪽 column – 성능이 다른 두 모델의 visualization 결과

→ 모델 간 ImageNet classification 성능 차이가 4~5% 정도가 있고, COCO2017 detection 성능 차이도 6~7%가 된다면

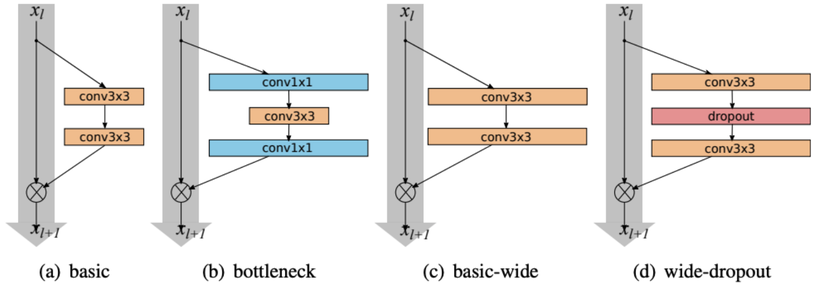
→ 결국 비행기, 사람, 연을 모두 찾아낼 수 있게 될 것입니다

DEVIEW
2019

3. 최신/유명한 가벼운 딥러닝 모델들 소개와 몇가지 이슈

3.0 무거운 (뚱뚱한) 모델들

DEVIEW 2019



(a) basic (b) bottleneck (c) basic-wide (d) wide-dropout

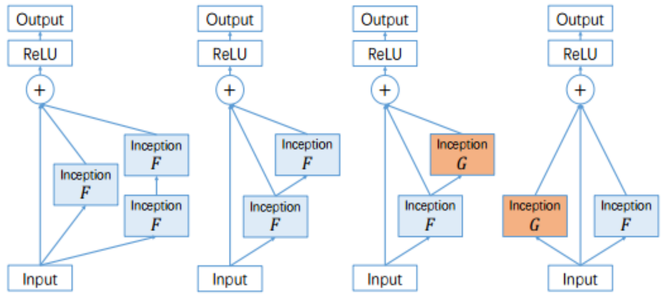
WideResNet (arXiv 2016)

ResNet+ Inception (Inception V4, arXiv 20

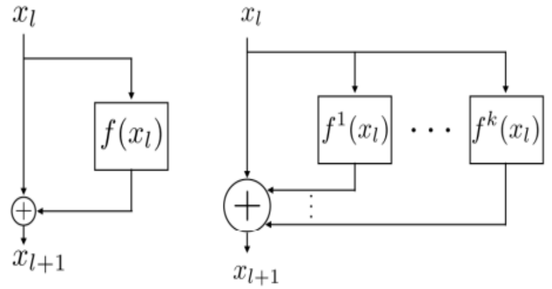
Multi-ResNet (arXiv 2017)

ResNext (**ResNet-V3**, CVPR 2017)

PolyNet (CVPR 2017)



(a) *poly-2* (b) *poly-2* (c) *mpoly-2* (d) *2-way*



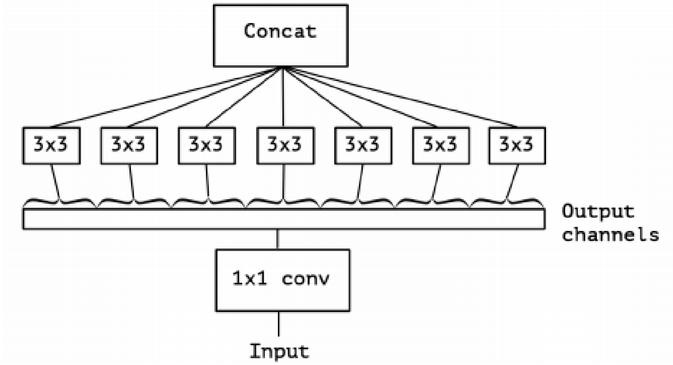
PyramidNet (CVPR2017)

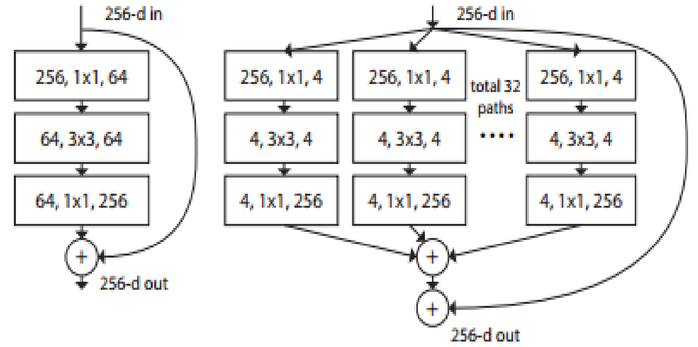
Xception (CVPR 2017)

DenseNet (arXiv 2016, 2017(v2))

DPN

SE-Net





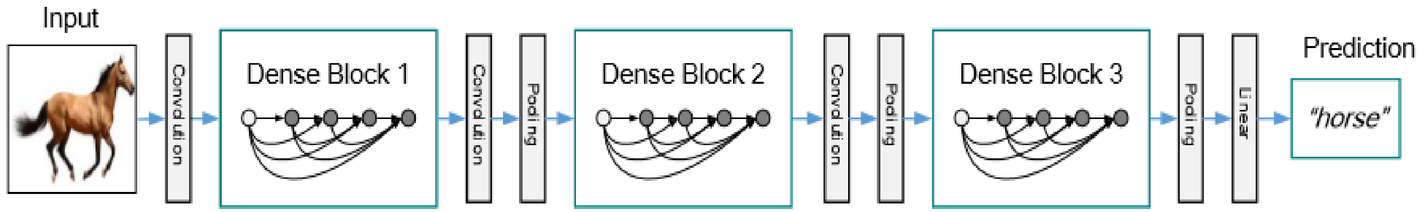
AmoebaNet

NasNet

PNas-Net

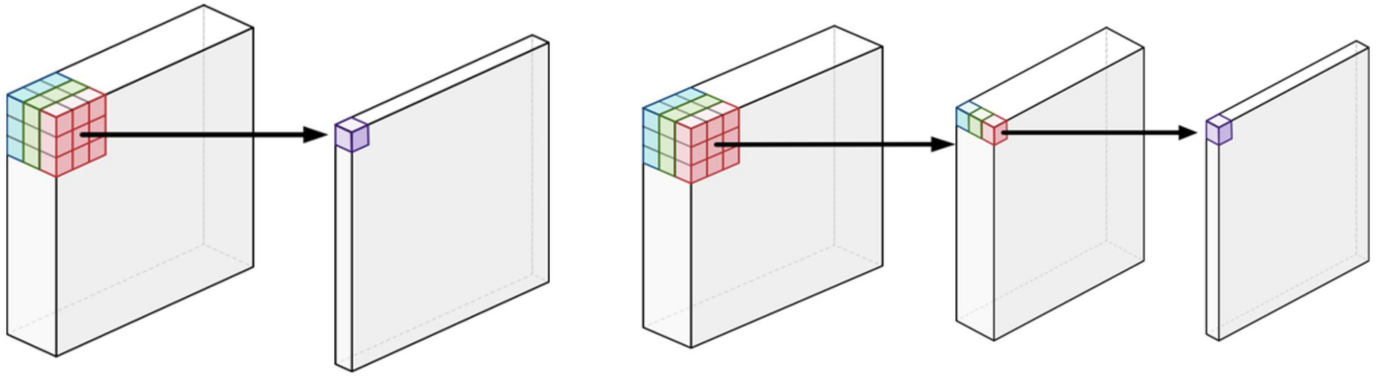
.

.



3.0 최신 가벼운 모델들

- 모델 개발 역사) 무거운 모델에서 가벼운 모델로 넘어갈 때 중요한 operator가 개발 되었습니다
  → (depthwise) separable convolution의 개발 (MobileNetV1 논문에서 첫 등장!)
  → 연산의 효율성을 비교해 보면 parameter 수가 (b)가 월등히 적습니다:
    - (a) regular conv.: 27 = 3x3x3x1
    - (b) separable conv.: 12 = 3x3 + 1x1x3x1 (depthwise conv.와 pointwise conv. 각각 계산)

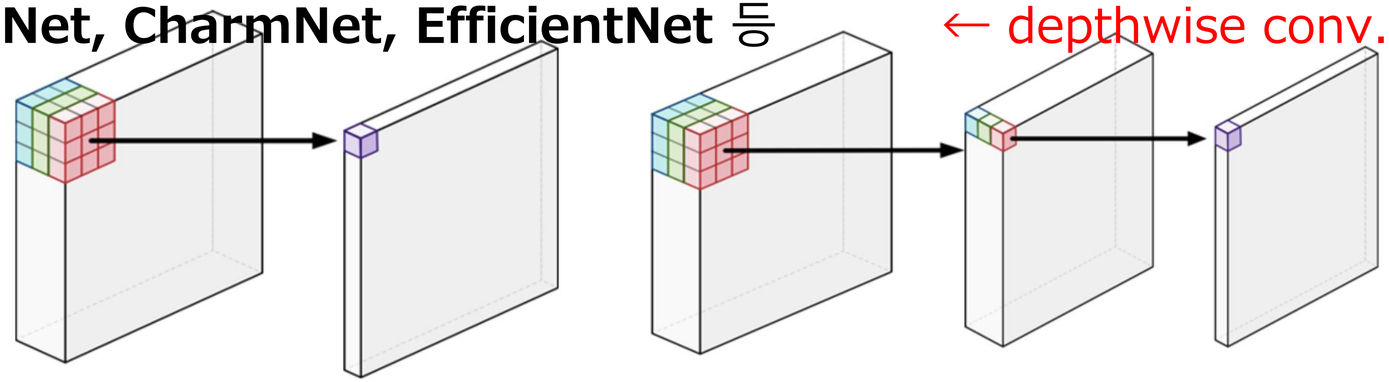


(a) regular convolution (b) depthwise separable convolution

이미지 출처: https://ai-diary-by-znreza.com/object-detection-with-single-shot-multibox-detector

3.0 최신 가벼운 모델들

- 모델 개발 역사) 무거운 모델에서 가벼운 모델로 넘어갈 때 중요한 operator가 개발 되었습니다
- 유명한 모델들:
  - → SqueezeNet,
  - → **MobileNetV1, MobileNetV2, MobileNetV3** ← depthwise conv. 사용
  - → **ShuffleNetV1, ShuffleNetV2** ← depthwise conv. 사용
  - → **MNasNet, FBNet, CharmNet, EfficientNet** 등 ← depthwise conv. 사용



(a) regular convolution (b) depthwise separable convolution

이미지 출처: https://ai-diary-by-znreza.com/object-detection-with-single-shot-multibox-detector

3.1 최신 가벼운 모델들의 성능 재현

DEVIEW 2019

- 논문을 잘 따라 ImageNet에 트레이닝 시켜도 리포트된 성능을 얻기 쉽지 않습니다
  예) MobileNetV2x1.0 모델, ShuffleNetV2x1.4 등의 모델

- 이유1: 해당 논문들에서 중요 세팅들을 공개하지 않았기 때문
- 이유2: 다수의 GPU (수십 개)를 한번에 써서 트레이닝 하기 쉽지 않기 때문

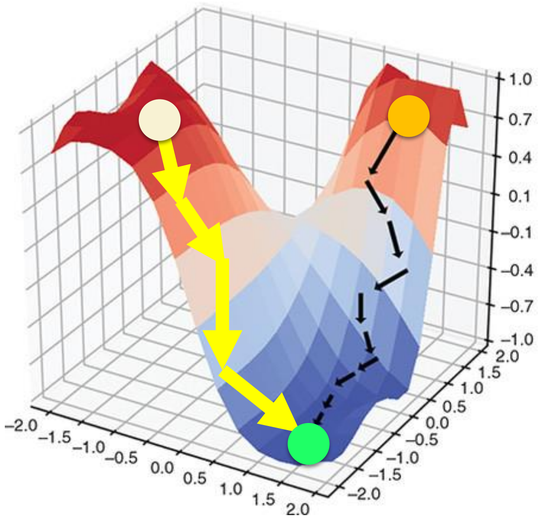
3.1 최신 가벼운 모델들의 성능 재현

DEVIEW 2019

- 논문을 잘 따라 ImageNet에 트레이닝 시켜도 리포트된 성능을 얻기 쉽지 않습니다
  예) MobileNetV2x1.0 모델, ShuffleNetV2x1.4 등의 모델

- 저희는 더 좋은 모델을 개발하려 했기 때문에 위 baseline들의 성능을 꼭 재현해야만 했습니다. - 논문과는 다른 트레이닝 세팅을 찾아냄으로 해결했습니다
- 즉 논문의 세팅대로 트레이닝했는데 결과가 도저히 안 나온다면?
  - 다른 세팅으로 동일 결과를 얻을 수 있으므로
    트레이닝 세팅 스터디를 해야합니다

※ 트레이닝 세팅 – optimizer 종류, learning rate 값, weight decay 값, epoch 수,
learning rate scheduling 방법,
data augmentation 방법/강도, 기타 regularizer 유/무



이미지 출처: https://livebook.manning.com/book/deep-learning-and-the-game-of-go/chapter-5/139

3.2 특정 모델들의 finetuning 성능 이슈

- 모델의 ImageNet 성능은 좋으나, finetuning 성능이 그만큼 좋지 않습니다:
  → 예) MobileNetV2, MobileNetV3, MNasNet

Backbone	mAP	Latency (ms)	Params (M)	MAdds (B)
V1	22.2	228	5.1	1.3
V2	22.1	162	4.3	0.80
MnasNet	23.0	174	4.88	0.84
V3	22.0	137	4.97	0.62
V3†	22.0	119	3.22	0.51

표 출처: https://arxiv.org/pdf/1905.02244.pdf

3.3 실제 서비스 이슈 – 성능 측면

- 실제 서비스에 backbone들을 적용하기 위해서 고려해야할 요소
  - 성능: backbone의 성능과 finetuning 후 해당 task의 성능
  - 속도: 총 flops (operator들의 총 연산량) 에 속도가 비례하는가?
  - 모델 크기: 모델 파라미터 수와 분포에 따라 training 또는 test 시 GPU 메모리를 얼마나

점유할지 결정, 즉 한번에 serving할 batchsize에 영향을 줍니다

3.3 실제 서비스 이슈 – 성능 측면

- 실제 서비스에 backbone들을 적용하기 위해서 고려해야할 요소
  - 성능: backbone의 성능과 finetuning 후 해당 task의 성능
  - 속도: 총 flops (operator들의 총 연산량) 에 속도가 비례하는가?
  - 모델 크기: 모델 파라미터 수와 분포에 따라 training 또는 test 시 GPU 메모리를 얼마나
    점유할지 결정, 즉 한번에 serving할 batchsize에 영향을 줍니다

- **Q) 성능이 무조건 좋은 backbone 을 택하면 되는가?**
  (A) YES, 단 Flops + 파라미터 수 사이는 대체적으로 비례하므로 이에 따라 적절한 backbone 선택

3.3 실제 서비스 이슈 – 속도 측면

- 실제 서비스에 backbone들을 적용하기 위해서 고려해야할 요소
  - 성능: backbone의 성능과 finetuning 후 해당 task의 성능
  - 속도: 총 flops (operator들의 총 연산량) 에 속도가 비례하는가?
  - 모델 크기: 모델 파라미터 수와 분포에 따라 training 또는 test 시 GPU 메모리를 얼마나

점유할지 결정, 즉 한번에 serving할 batchsize에 영향을 줍니다

+ 성능과 모델 크기와 달리 속도와 flops이 환경에 따라 비례하지 않는 경우가 종종 있습니다

3.3 실제 서비스 이슈 – 속도 측면

DEVIEW 2019

- **Q) 모델의 Flop이 작으면 속도도 빨라지는가?**

  A) 대체적으로 비례하나 꼭 그렇지도 않습니다 (아래 예 참고):

  - Concatenation이나 depthwise-conv등은 플랫폼에 따라, 하드웨어에 따라 속도가 다름
    - Flops은 0이나 실제 속도가 느린 Reshape, AveragePool, Upsample 등의 operator들 존재:

+ 단, GPU에서는 현 구현상 이슈로 regular convolution을 쓰는 것이 depthwise convolution이나 group convolution을 쓰는 것보다 훨씬 빠릅니다

+ 그래서 ResNeXt (ResNetV3) 나 MobileNetV2가 GPU에서 FLOPs 대비 느립니다

3.3 실제 서비스 이슈 – 속도 측면

- **Q) 모델의 Flop이 작으면 속도도 빨라지는가?**
  - A) 대체적으로 비례하나 꼭 그렇지도 않습니다 (아래 예 참고):
    - Concatenation이나 depthwise-conv등은 플랫폼에 따라, 하드웨어에 따라 속도가 다름
      - Flops은 0이나 실제 속도가 느린 Reshape, AveragePool, Upsample 등의 operator들 존재:

Model	Complexity (MFLOPs)	Top-1 err. (%)	GPU Speed (Batches/sec.)	ARM Speed (Images/sec.)
ShuffleNet v2 0.5× (ours)	41	**39.7**	417	**57.0**
0.25 MobileNet v1 [13]	41	49.4	**502**	36.4
0.4 MobileNet v2 [14] (our impl.)*	43	43.4	333	33.2
0.15 MobileNet v2 [14] (our impl.)	39	55.1	351	33.6
ShuffleNet v1 0.5× (g=3) [15]	38	43.2	347	56.8
DenseNet 0.5× [6] (our impl.)	42	58.6	366	39.7
Xception 0.5× [12] (our impl.)	40	44.9	384	52.9
IGCV2-0.25 [27]	46	45.1	183	31.5

표 출처: https://arxiv.org/pdf/1807.11164.pdf

3.3 실제 서비스 이슈 – 모델 크기 측면

DEVIEW 2019

- 실제 서비스에 backbone들을 적용하기 위해서 고려해야할 요소
  - 성능: Accuracy, precision/recall등
  - 속도: Flops (operator들의 총 연산량) 에 속도가 비례하는가?
  - 모델 크기: 모델 파라미터 수와 분포에 따라 training 또는 test 시 GPU 메모리를 얼마나

점유할지 결정, 즉 한번에 serving할 batchsize에 영향을 줍니다

+ 모델 크기는 허용하는 만큼 크게 (성능을 극대화) 선택하는 것이 좋습니다
+ 그러나 모델 크기가 크다고 속도가 반드시 느려지는 것은 아닙니다

DEVIEW
2019

4. 새로 개발한 가벼운 모델과 성능 소개

4.0 모델 설계 배경

- 현재 가벼운 모델들 사이의 이슈:
  - ImageNet 성능은 좋으나, finetuning 성능이 상대적으로 떨어지는 문제
    (예: MobileNetV3, MNasNet, MixNet)
  - 현재 SOTA인 대부분의 모델 전부가 NAS로 찾아낸 모델들이며 ImageNet 분류 문제에
    fitting 되어 설계된 모델

4.0 모델 설계 목적 & 방법

- 설계 목적:
  - 기존 모델들의 구조 상의 문제를 해결해보자
  - ImageNet 데이터셋에서 가벼운 구조 중에 SOTA를 얻어보자
  - CPU inference 용 모델 개발 (단, 개발된 모델은 GPU에서도 느리지 않습니다)
  - Finetuning이 잘되는 구조 개발 (transferability 극대화 목표)
  - NAS 모델이나 NetAdapt 류 method의 baseline 모델로 제공하자

- 설계 방법:
  - 성능 bottleneck을 해결하고 나서 모델을 축소하여 flop과 실 속도를 확보하자
  - Flop-efficient한 부분들의 feature 수가 부족하지 않도록 구조를 개선

4.1 모델 설계 detail

- 모델 구조 개선
  - 가정: Finetuning이 잘 안되는 이유는 지나치게 flop-efficient 한 구조를 만들어 냈기 때문 (예: MobileNetV2, MobileNetV3, MNasNet)

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	-	

MobileNetV2 layer 구조와 단면도

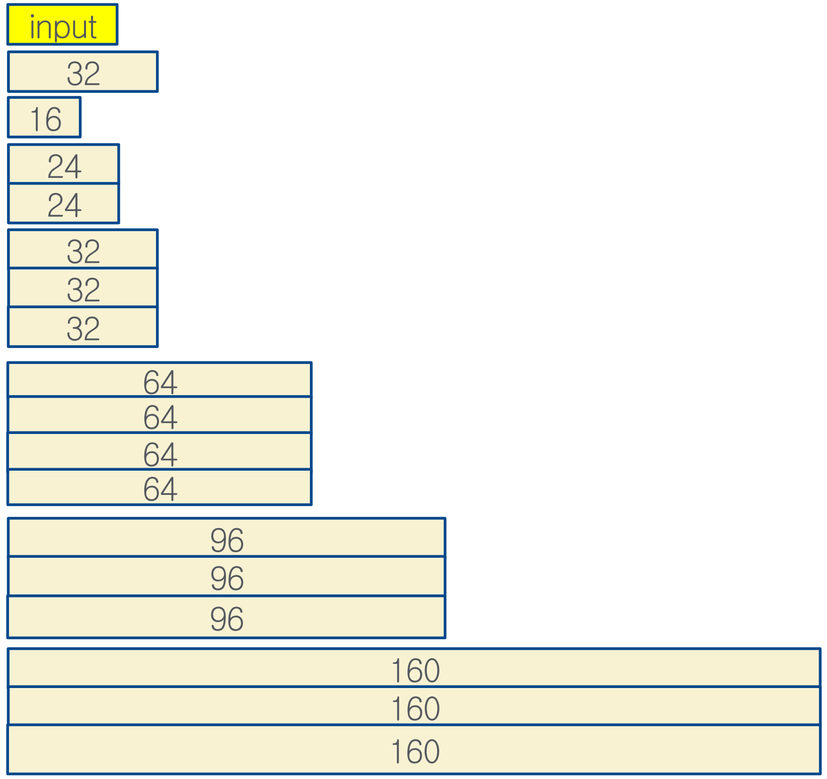


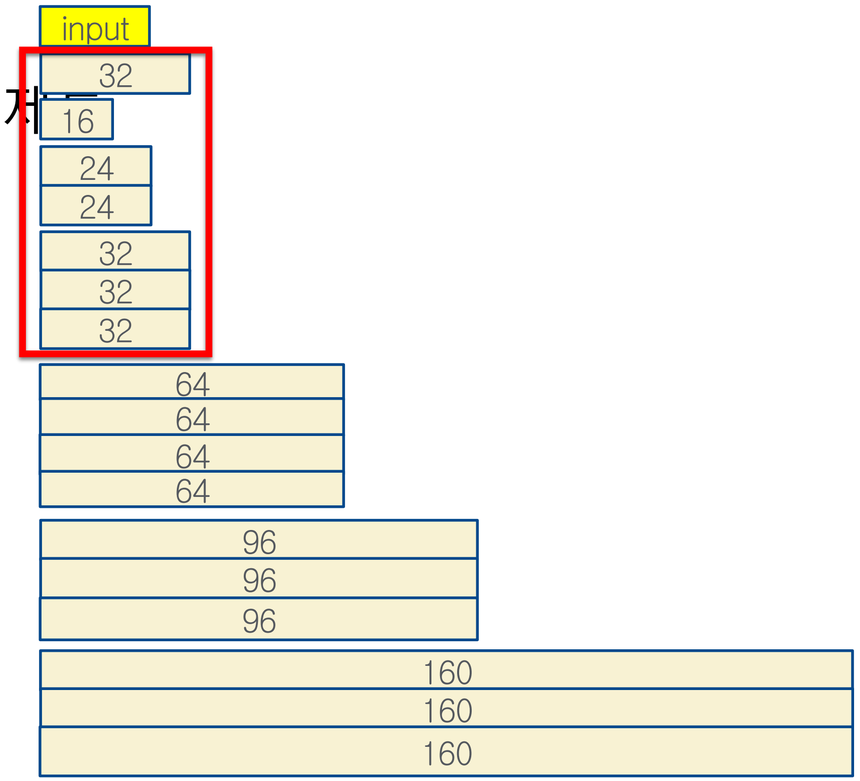
표 출처: https://arxiv.org/pdf/1801.04381.pdf

4.1 모델 설계 detail

DEVIEW 2019

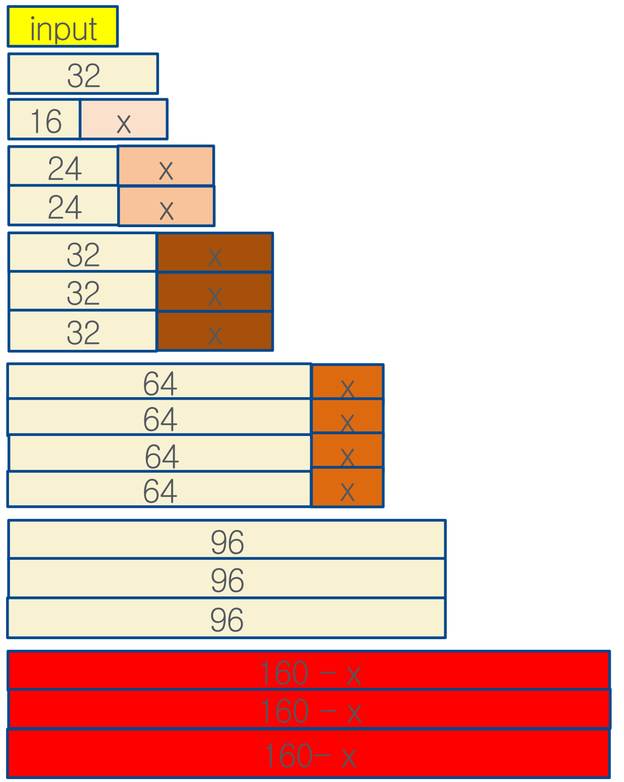
- 모델 구조 개선
  - 아래 빨간 박스 안의 layer에서 feature 수가 지나치게 부족한것처럼 보입니다
    (속도를 생각해서 flop-efficient 하게 설계하려고 input쪽 flops을 줄인 것입니다)

  - 이러한 layer 들이 finetuning 시에 성능 저하 문제를
    일으킬 것으로 추정



4.1 모델 설계 detail

- 모델 구조 개선
  - 이렇게 모델을 바꾸면 어떨까?
  - 특정 레이어 (output-side)의 feature 수를 줄이고 부족한 layer의 feature 수 보강
  - 보강 방법은 **여전히 open-question 이므로**
    하나의 세팅을 찾고 성능으로 conjecture을 증명



4.2 모델 설계 프로세스 요약

1) 기존 모델 성능 reproduction - 제일 중요합니다
2) 비교 모델들 성능 reproduction
3) 신 모델 연구 (설계/트레이닝/테스트 반복)
   - 모델 설계를 계속해서 바꿔가며 성능을 측정합니다
   - ImageNet-1k이 매우 크므로 작은 데이터셋 (예: cifar 등)에서 먼저 실험을 추천합니다.
4) 모델 트레이닝 방법 (training scheme) 연구/최적화 – 최적 세팅을 연구하는 것이 중요합니다
5) 모델 정밀 설계 – 추가 모듈을 통한 성능 향상 여지가 더 있는 확인합니다
6) Finetuning 결과 확보 – object detection이나 segmentation 등에 활용하여 성능 확인

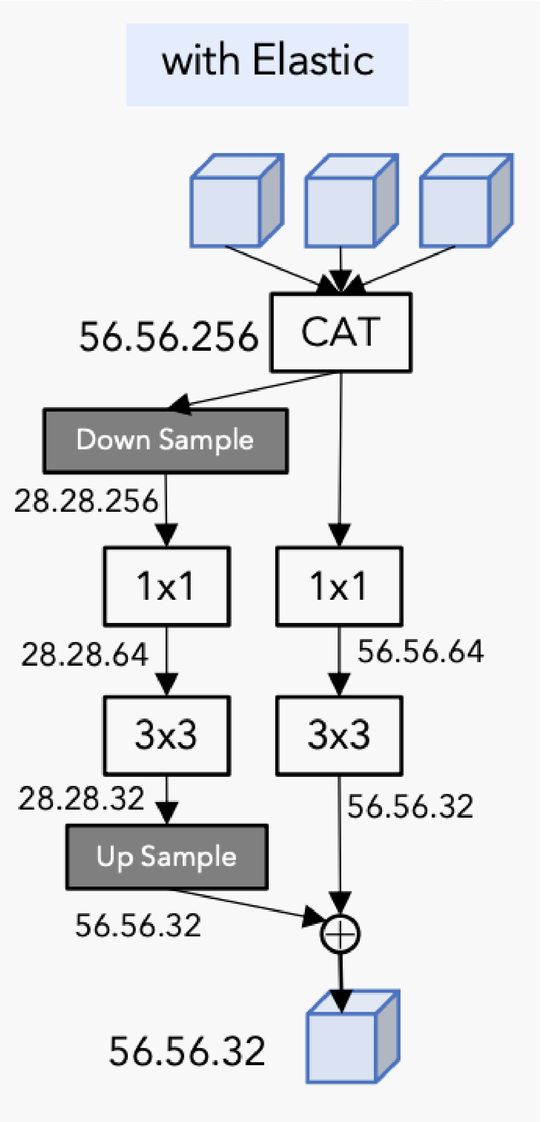
4.3 신 모델에 쓰인 약간의 detail

DEVIEW 2019

- MobileNetV1, V2 베이스로 개발
- Data augmentation은 ResNet과 동일
- SE-net 사용 (기존 NAS 모델과 fair comparsion 위해서)

+ Flop을 줄일 수 있지만 가벼운 모델에서 속도가 느린 것들은
  - Pooling류, Upsample류, 때로는 concat도 느리므로 배제
  - ELASTIC (OctConv도 마찬가지) 배제

ELASTIC 모듈



이미지 출처: https://arxiv.org/abs/1805.08974

4.3 신 모델의 학습 환경

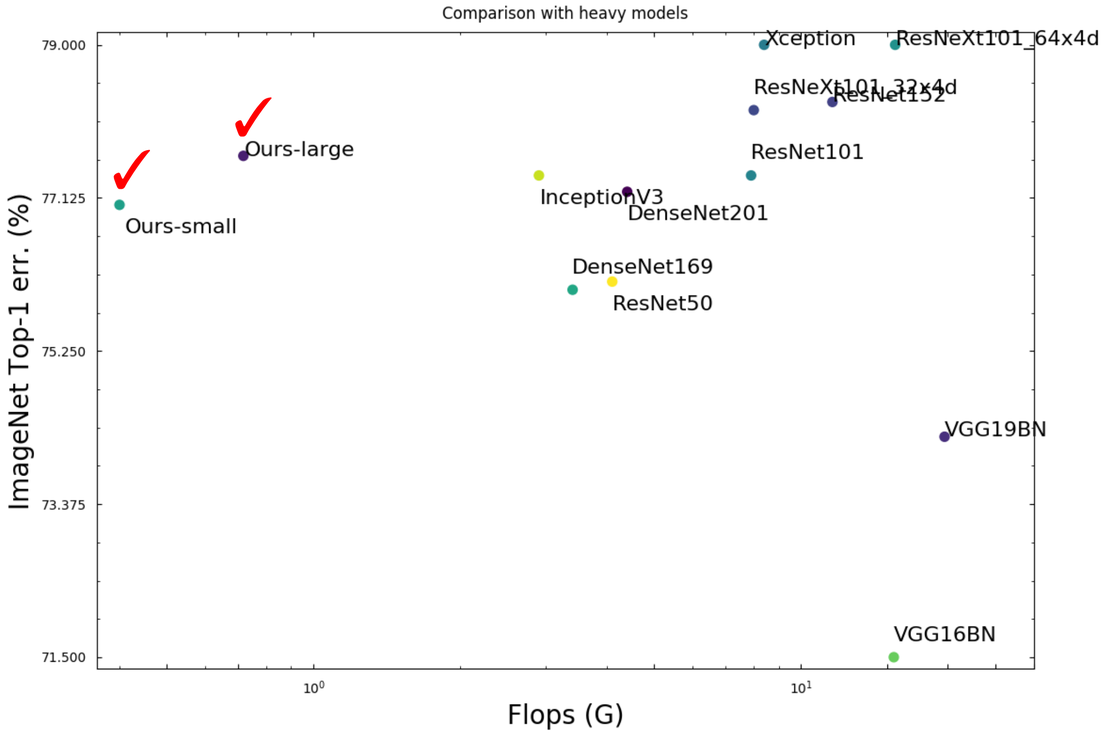
- Training 세팅 (MobileNet 등 기본 training 세팅에 더하여):
  - Mixed precision (fp16 + fp32) 사용
  - Learning rate 등 training hyper parameter들의 최적 세팅 경험적 튜닝
  - Tensorflow 기본 세팅인 EMA (exponential moving average) 등 사용하지 않음

- Test 세팅:
  - 기존 모델과 동일한 standard setting (256 resize 후 224x224 center crop)
  - 공정한 비교를 위해 찾아낸 세팅들 비교 모델에 모두 적용 후 비교

4.3 ImageNet 성능 vs 무거운 모델

DEVIEW 2019

Model	Top-1 err	Top-5 err	Params	Flops
Ours-large	22.36	6.16	3.99M	0.71G
Ours-small	22.96	6.80	3.76M	0.40G
VGG16BN	28.5	10.2	138.4M	15.5G
VGG19BN	25.8	8.2	143.7M	19.7G
ResNet50	23.9	7.1	23.5M	4.1G
ResNet101	22.6	6.4	42.5M	7.9G
ResNet152	21.7	5.9	60.2M	11.6G
InceptionV3	22.6	6.4	27.2M	2.9G
RsnNeXt101_32x4d	21.8	6.1	44.2M	8.0G
ResNeXt101_64x4d	21	5.8	83.5M	15.6G
DenseNet169	24	7	14.2M	3.4G
DenseNet201	22.8	6.4	20.0M	4.4G
Xception	21	5.5	22.9M	8.4G

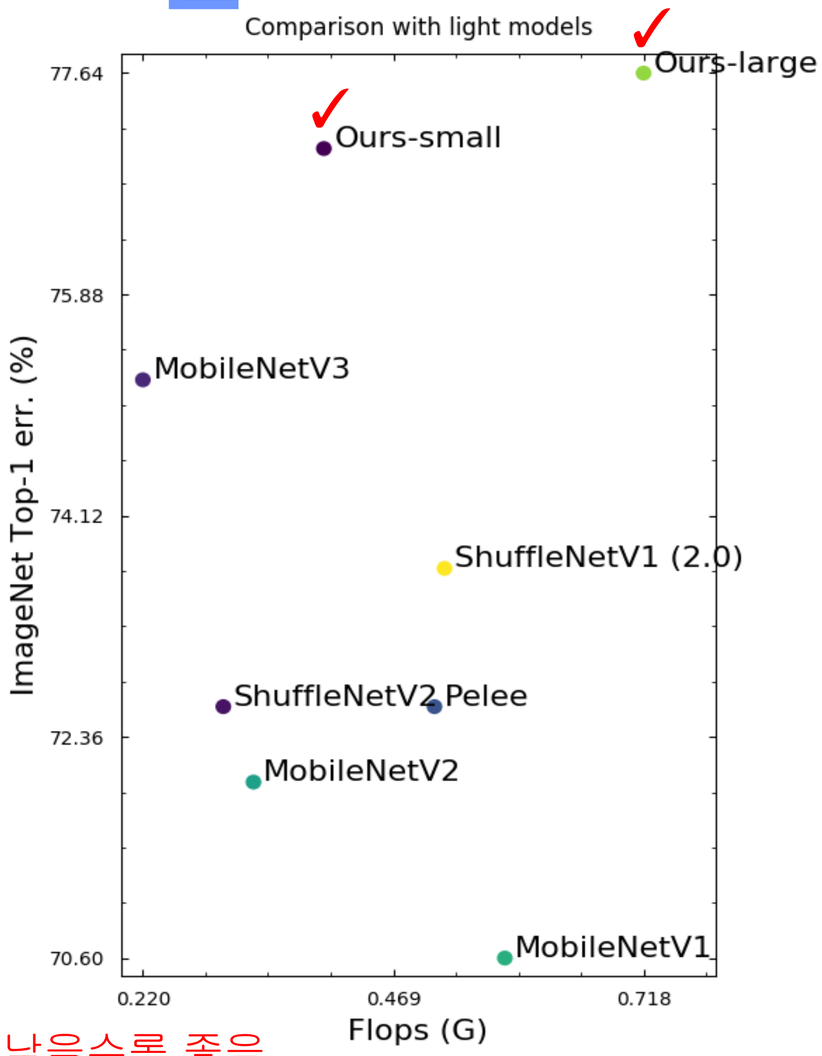


- Top-1 err. (%): 모델이 예측한 최상위 class 가 틀린 이미지의 비율 -> 낮을수록 좋음
- Top-5 err. (%): 모델이 예측한 상위 다섯개의 class 중 정답이 없는 이미지 비율 -> 낮을 수록 좋음
- Params, flops: 총 learnable weight 파라미터 수, 총 연산량

4.3 ImageNet 성능 vs 가벼운 모델

DEVIEW 2019

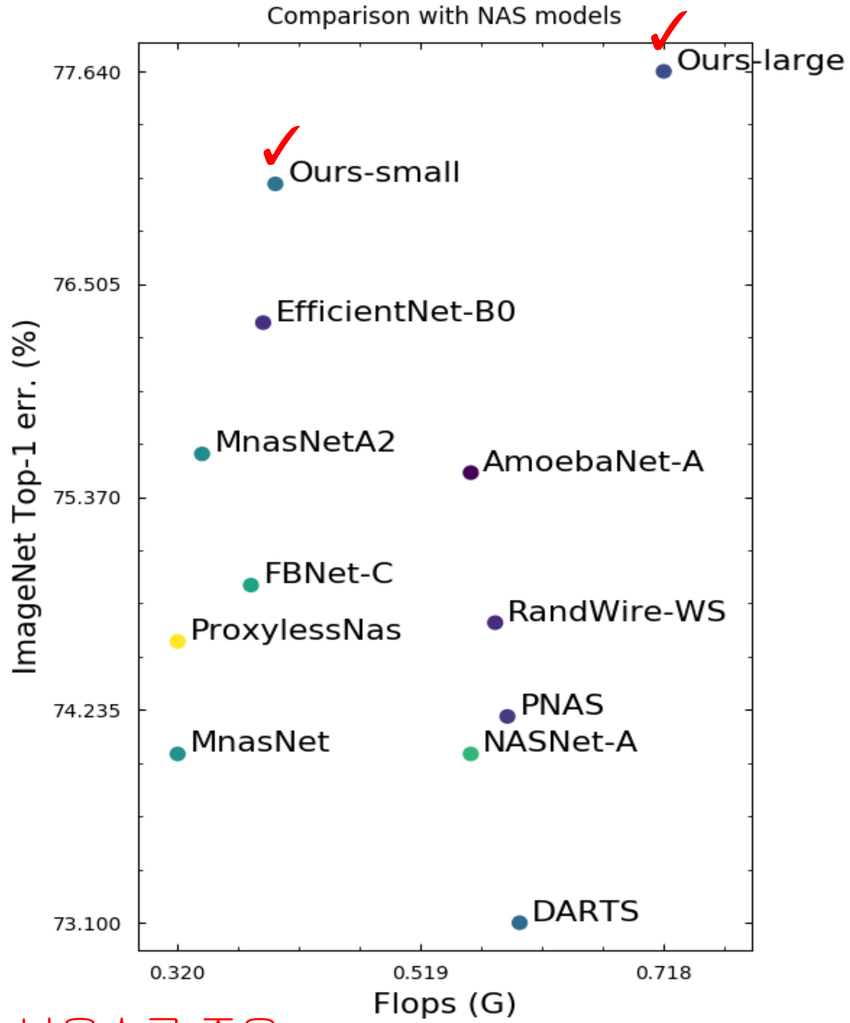
Model	Top-1 err	Top-5 err	Params	Flops
Ours-large	22.36	6.16	3.99M	0.71G
Ours-small	22.96	6.80	3.76M	0.40G
MobileNetV1	29.4	10.5	4.2M	0.58G
MobileNetV2	28.0	9.0	3.5M	0.33G
MobileNetV3-Large	24.8	-	5.4M	0.22G
CondenseNet	26.2	8.3	4.8M	0.53G
ShuffleNetV1 (2.0)	26.3	-	5.4M	0.52G
ShuffleNetV2	27.4	-	3.5M	0.30G
Pelee	27.4	9.4	2.8M	0.51G



- Top-1 err. (%): 모델이 예측한 최상위 class 가 틀린 이미지의 비율 -> 낮을수록 좋음
- Top-5 err. (%): 모델이 예측한 상위 다섯개의 class 중 정답이 없는 이미지 비율 -> 낮을 수록 좋음
- Params, flops: 총 learnable weight 파라미터 수, 총 연산량

4.3 ImageNet 성능 vs NAS 모델들

Model	Top-1 err	Top-5 err	Params	Flops
Ours-large	22.36	6.16	3.99M	0.71G
Ours-small	22.96	6.80	3.76M	0.40G
NASNet-A	26	8.3	5.3M	0.56G
AmoebaNet-A	24.5	8	5.1M	0.56G
PNAS	25.8	8.1	5.1M	0.59G
DARTS	26.9	9	4.9M	0.60G
FBNet-C	25.1	-	5.5M	0.38G
ProxylessNas	25.4	6.7	4.1M	0.32G
RandWire-WS	25.3	7.8	5.6M	0.58G
MnasNet	26	8.2	4.2M	0.32G
MnasNetA2	24.4	7.3	4.8M	0.34G
EfficientNet-B0	23.7	6.8	5.3M	0.39G



- Top-1 err. (%): 모델이 예측한 최상위 class 가 틀린 이미지의 비율 -> 낮을수록 좋음
- Top-5 err. (%): 모델이 예측한 상위 다섯개의 class 중 정답이 없는 이미지 비율 -> 낮을 수록 좋음
- Params, flops: 총 learnable weight 파라미터 수, 총 연산량

DEVIEW
2019

5. 신 모델을 활용한 finetuning 성능과 실 서비스 적용 예 + 추가 팁

5.1 Object detection 적용 결과

DEVIEW 2019

- 개발한 신 모델을
  SSD-lite라는 가벼운 detection 모듈과 결합
- 비교대상 (아래는 모두 최신 SOTA 모델들):
  - MobileNetV1 + SSD-lite
  - MobileNetV2 + SSD-lite
  - MNasNet-A1 + SSD-lite
  - MobileNetV3 + SSD-lite
  - MixConv + SSD-lite
- 우리 모델의 성능- mAP (%) 이 월등히 좋음

Model	mAP (%)	Params	Flops	latency	Img. Res.
Ours-large + SSD-lite	26.5	4.88M	1.66G	32.3ms	320x320
Ours-small + SSD-lite	25.4	4.65M	1.02G	31.7ms	320x320
MobileNetV1 + SSD-lite	22.2	5.1M	1.31G	30.8ms	320x320
MobileNetV2 + SSD-lite	22.1	4.3M	0.79G	31.4ms	320x320
MobileNetV3 + SSD-lite	22.0	4.97M	0.62G	-	320x320
MNasNet-A1 + SSD-lite	23.0	4.9M	0.84G	-	320x320
MobV1+mixconv + SSD-lite	22.4	5.22M	1.39G	-	320x320
MobV2+mixconv + SSD-lite	22.3	4.49M	0.88G	-	320x320

5.2 Scene text detection 적용 결과

DEVIEW 2019

- Scene text detection (텍스트 검출) 에서 SOTA 결과를 유지하고 있는 CRAFT (CVPR 2019)

Character Region Awareness for Text Detection

Youngmin Baek, Bado Lee, Dongyoon Han, Sangdoo Yun, and Hwalsuk Lee*
Clova AI Research, NAVER Corp.
{youngmin.baek, bado.lee, dongyoon.han, sangdoo.yun, hwalsuk.lee}@navercorp.com

Abstract

Scene text detection methods based on neural networks have emerged recently and have shown promising results. Previous methods trained with rigid word-level bounding boxes exhibit limitations in representing the text region in an arbitrary shape. In this paper, we propose a new scene text detection method to effectively detect text area by exploring each character and affinity between characters. To overcome the lack of individual character level annotations, our proposed framework exploits both the given character-level annotations for synthetic images and the estimated character-level ground-truths for real images acquired by the learned interim model. In order to estimate affinity between characters, the network is trained with the newly proposed representation for affinity. Extensive experiments on six benchmarks, including the TotalText and CTW-1500 datasets which contain highly curved texts in natural images, demonstrate that our character-level text detection

Figure 1. Visualization of character-level detection using CRAFT.

5.2 Scene text detection 적용 결과

- 모델의 backbone 부분을 우리 모델로 대체하고 classifier 부분도 최대한 경량화 (CRAFT-lite)

- 파라미터는 1/9 수준으로 줄이고 Hmean은 그대로!

Model	Backbone	ImageNet Top1 err (%)	Hmean	params
CRAFT	VGG16BN	26.6%	91.5%	20.8M
CRAFT-lite	ReM1.2.p2	25.4%	**91.0%**	2.3M

- 단 사용된 모델은 legacy 버전 (25.4% err.), 즉 최신 버전 (22.4% err.)을 쓴다면?
  → 성능 대폭 향상이 예상됩니다

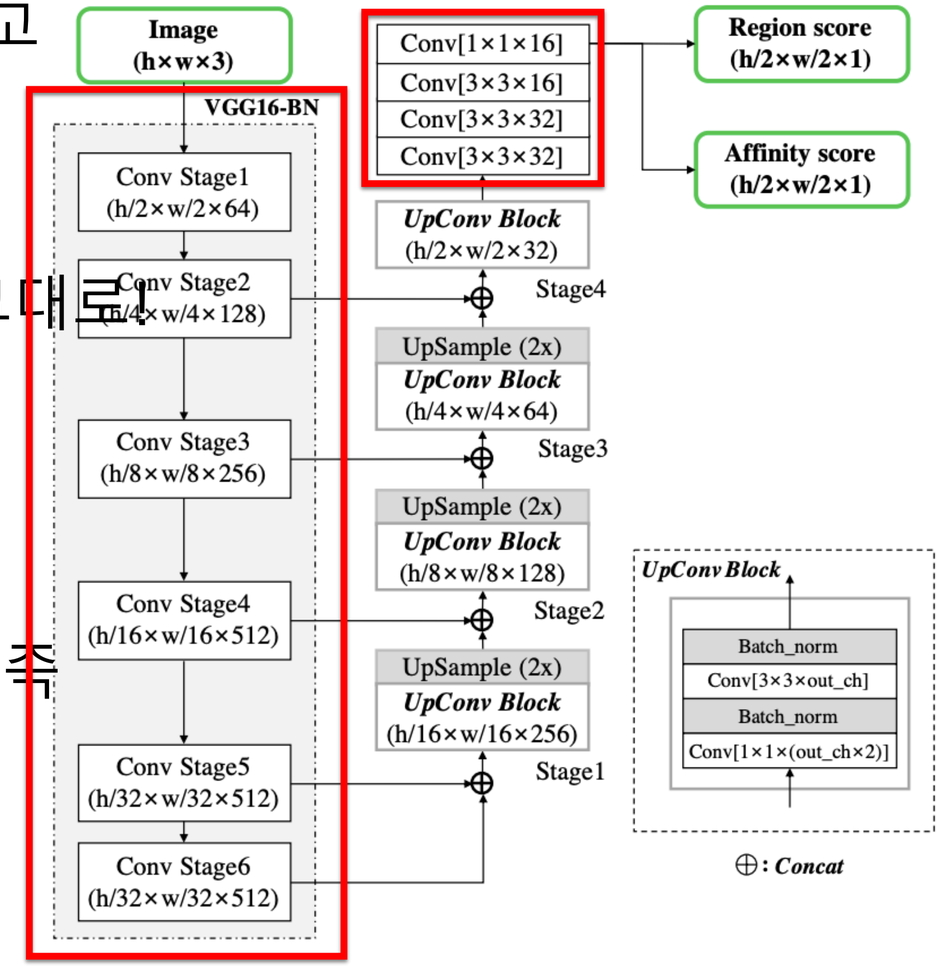
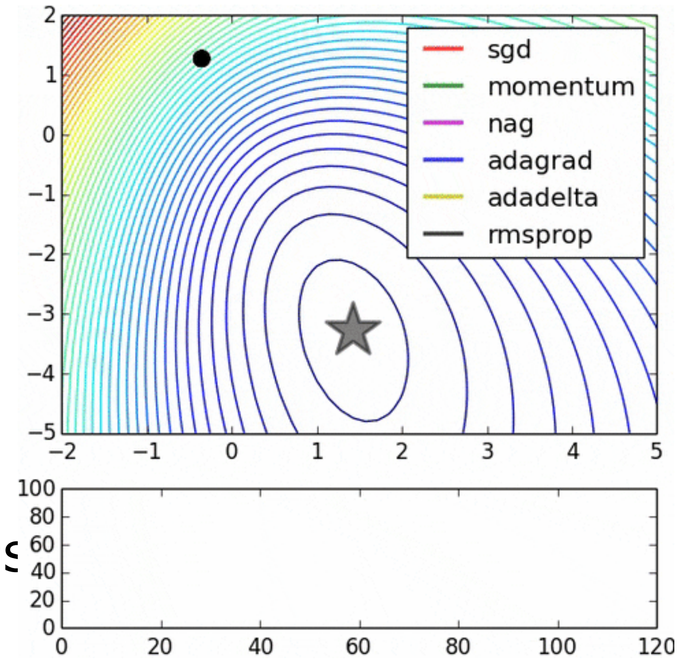


Figure 2. Schematic illustration of our network architecture.

이미지 출처: https://arxiv.org/pdf/1904.01941.pdf

5.3 팁: 가벼운 모델 트레이닝 시 유의점

DEVIEW 2019



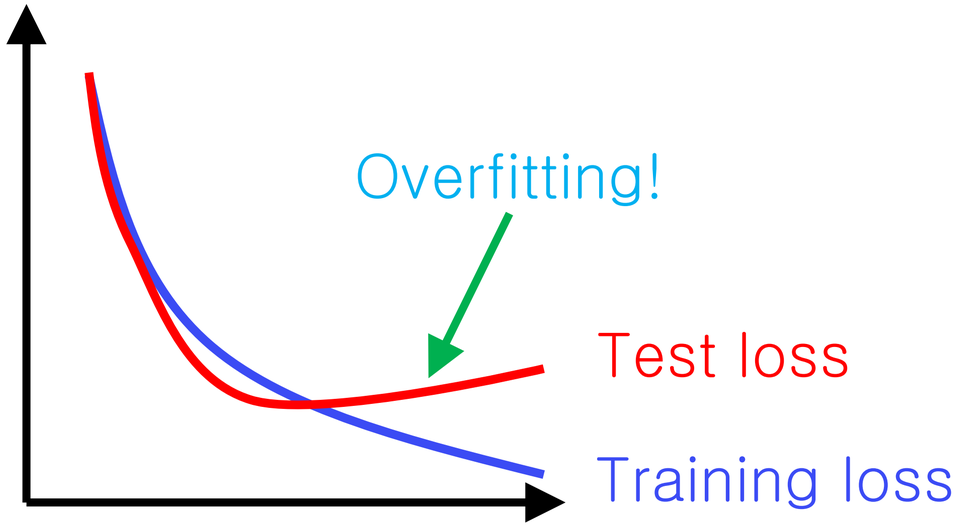
이미지 출처: https://media1.giphy.com/media/r4xVbqa6YLJu0/giphy.gif?cid=790b761169cd8466bc1e325548a41bcfadeebf500cce3296&rid=giphy.gif

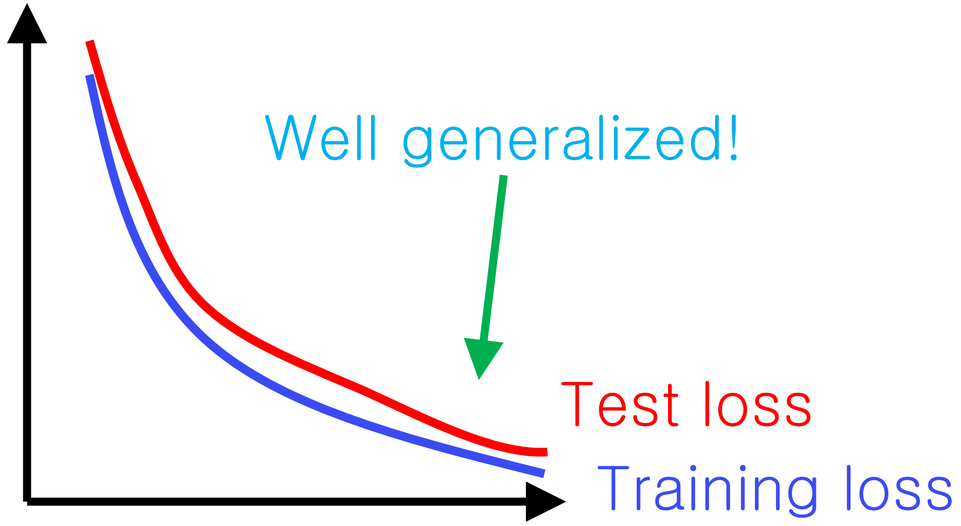
- 가벼운 모델 트레이닝법은 기존 무거운 모델과 다소 다를 수 있습니다
  → 이것이 reproducing에 실패하는 이유

- 유의점:
  - ADAM류 (ADAM, RADAM 등) 보다 SGD가 훨씬 좋습니다 (분류 tas
    (Nesterov SGD+momentum이 가장 좋습니다)
  - SGD 트레이닝 파라미터들 스터디 필요 (특히 learning rate와 weight decay)
  - 트레이닝 커브를 볼 때 초반의 경향이 끝까지 유지되지 않을 수 있습니다
  - 트레이닝 커브를 **반드시** 매 트레이닝 시마다 저장해두어야 경향을 파악하고 대처할 수 있습니다
  - Data augmentation을 약하게 주어야 할 때가 있습니다

5.3 팁: 가벼운 모델 트레이닝 시 유의점

- 트레이닝 커브를 얻어야 하는 이유는 overfitting 여부를 반드시 파악해야하기 때문입니다
- 트레이닝 hyper-parameter 에 따라서 아래처럼 그래프의 경향이 다양하게 변합니다
  (특히 가벼운 모델은 초기에 트레이닝 잘되는 것 처럼 보여도 후반에 역전되는 등 다양한 경향을 보입니다)





5.3 팁: 기존 모델 성능 reproduction 방법

- 팁을 적용하고 나면,
  - 즉, 트레이닝을 더 잘하면 아래와 같이 기존 모델 성능도 향상 시킬 수 있습니다:

Model	Top-1 err	Top-5 err	Params	Flops	Desc.	Model
MobileNetV1 (ours)	27.46	9.31	4.2M	0.58G	our result	-
MobileNetV2 (ours)	26.96	8.76	3.51M	0.33G	our result	-
MobileNetV1	29.4	10.5	4.2M	0.58G	paper's	-
MobileNetV2	28.0	9.0	3.5M	0.33G	paper's	-

5.4 추가로 할 것: 강력한 Regularization 도입

DEVIEW 2019

- 모델의 성능이 확보되고 나면 해야할 것은 generalization 능력을 올리는 것
  - Data augmentation method 이자 regularization 방법이 필요,
- 최근에 공개된 **CutMix** (**ICCV 2019**) 이라는 현존 최고의 방법이 존재합니다
  - Code: https://github.com/clovaai/CutMix-PyTorch

CutMix: Regularization Strategy to Train Strong Classifiers with Localizable Features

Sangdoo Yun[1] Dongyoon Han[1] Seong Joon Oh[2] Sanghyuk Chun[1]
Junsuk Choe[1,3] Youngjoon Yoo[1]

[1]Clova AI Research, NAVER Corp.
[2]Clova AI Research, LINE Plus Corp.
[3]Yonsei University

Abstract

Regional dropout strategies have been proposed to enhance the performance of convolutional neural network classifiers. They have proved to be effective for guiding the model to attend on less discriminative parts of objects (e.g. leg as opposed to head of a person), thereby letting the network generalize better and have better object localization capabilities. On the other hand, current methods for regional dropout remove informative pixels on training images by overlaying a patch of either black pixels or random noise. Such removal is not desirable be-

	ResNet-50	Mixup [48]	Cutout [3]	CutMix
Image				
Label	Dog 1.0	Dog 0.5 Cat 0.5	Dog 1.0	Dog 0.6 Cat 0.4
ImageNet Cls (%)	76.3 (+0.0)	77.4 (+1.1)	77.1 (+0.8)	**78.6** **(+2.3)**
ImageNet Loc (%)	46.3 (+0.0)	45.8 (-0.5)	46.7 (+0.4)	**47.3** **(+1.0)**

	ResNet-50	Mixup	Cutout	CutMix
Image				
Label	Dog 1.0	Dog 0.5 Cat 0.5	Dog 1.0	Dog 0.6 Cat 0.4
ImageNet Cls (%)	76.3 (+0.0)	77.4 (+1.1)	77.1 (+0.8)	**78.4** **(+2.1)**
ImageNet Loc (%)	46.3 (+0.0)	45.8 (-0.5)	46.7 (+0.4)	**47.3** **(+1.0)**
Pascal VOC Det (mAP)	75.6 (+0.0)	73.9 (-1.7)	75.1 (-0.5)	**76.7** **(+1.1)**

6. 결론

6. 결론

- Backbone 모델의 성능이 목표 task 성능을 크게 좌우하므로 좋은 backbone을 써서 트레이닝 하는 것이 중요합니다
- 최신 공개된 모델들은 finetuning 시 성능이 만족스럽지 않을 수도 있습니다
- 그러므로 리소스가 충분하면 좋은 backbone 개발을 시도하시고, 그렇지 않으면 공개할 저희 모델을 쓰시면 됩니다 (ImageNet 성능과 Finetuning 성능까지 모두 좋습니다)

추가 소식

- Pretrained 모델 공개 예정입니다
- 논문 공개 예정입니다

DEVIEW
2019

감사합니다
Q & A

Appendix: 무거운 딥 러닝 모델 추천

모델 추천 1 (성능 측면):

1) VGG-16 (파라미터는 많으나, 생각보다 빠르고 성능이 좋음)

2) ResNet50-SE (ResNet101이나 ResNet152 대비 efficient, GPU 서빙의 마지노선)

3) ResNeXT101-SE + FPN (FPN은 detection등에 적용할 때 성능에 큰 영향)

4) Xception 계통 (의외로 효율적이고 강력한 모델, group-conv 기반)

5) EfficientNet-B4 (효율적이고 성능이 좋지만, dw-conv 기반)

+ AmoebaNet등의 NAS 기반 거대 모델들은 complexity나 scalability 이슈로 추천하지 않습니다

Appendix: 가벼운 딥 러닝 모델 추천

모델 추천 2 (속도 측면):

1) ResNet-18 (ResNet-50보다 빠르지만, 여전히 크기가 큼)
2) Xception 계통 (CPU용, Xception을 작게 만들어 사용)
3) MobileNetV1 (CPU용, MobileNetV2보다 더 좋을 때가 많음)
4) 저희 모델 (성능까지 우수 ☺)

+ 그나마 MobileNetV1이 성능 vs 속도 (+ 모델 크기) 면에서 좋은 편이지만, 문제는 finetuning (detection 등) 성능이 모델 크기 대비 좋다고 볼 수는 없습니다

+ 우리 모델은 속도도 빠르고 성능도 비교도 안되게 우수합니다