이미지 처리를 위한 모든 것

03. Advanced CNN, 다양한 CNN 구조

카이스트

김현우

0 본인 소개

김 현 우

카이스트, 산업및시스템공학과 대학원생

TEAM-EDA 블로그, 페이스북 페이지 운영

Recommender System KR 페이스북 그룹 운영진

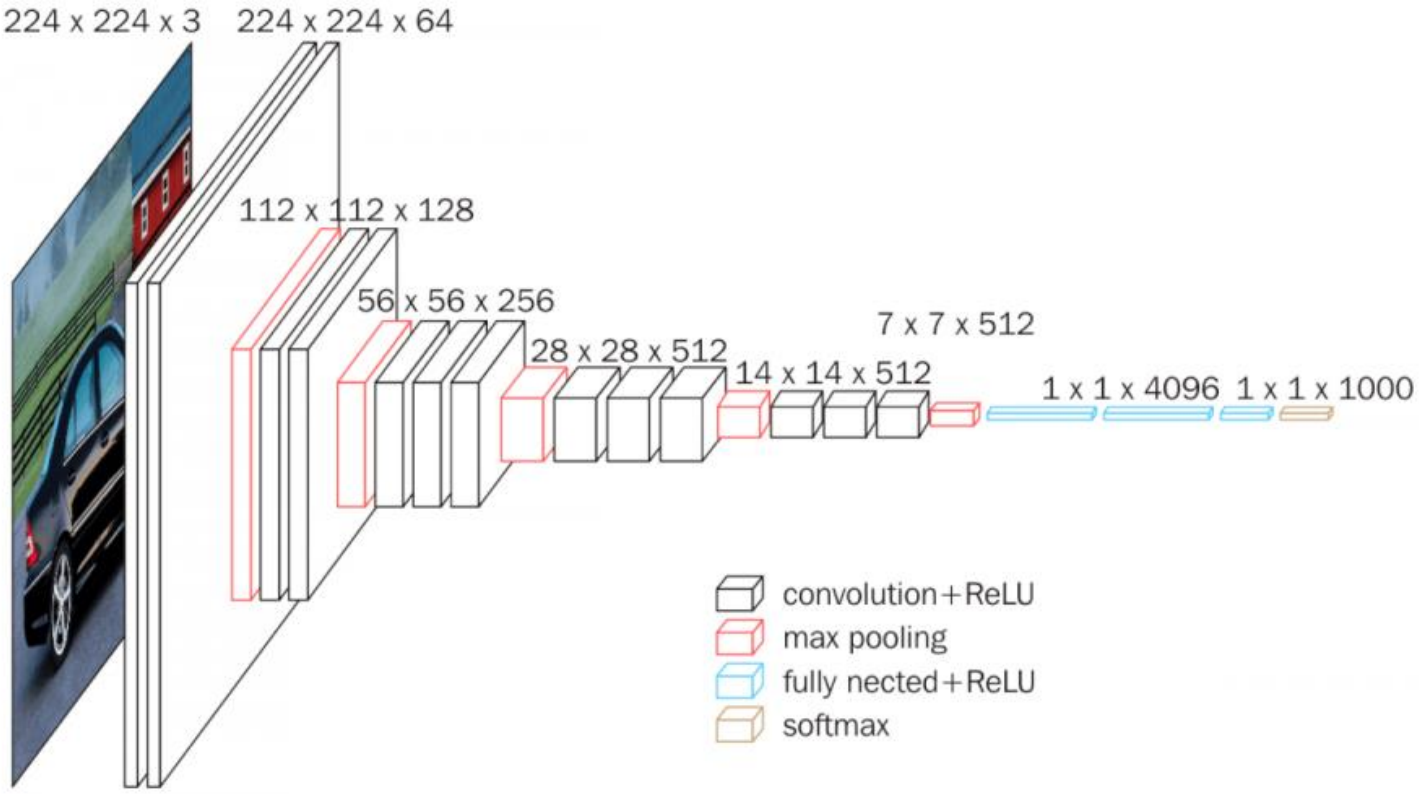
깃허브 : https://github.com/choco9966/

01 VGG

1 VGG

초록

VGG는 3X3의 Convolution filters만을 이용해서 층을 깊게 쌓아서 좋은 성능을 보인 모델입니다. 실제로 ImageNet Challenge2014와 localization and classification tracks에서 상위권에 랭크했습니다.



1 VGG

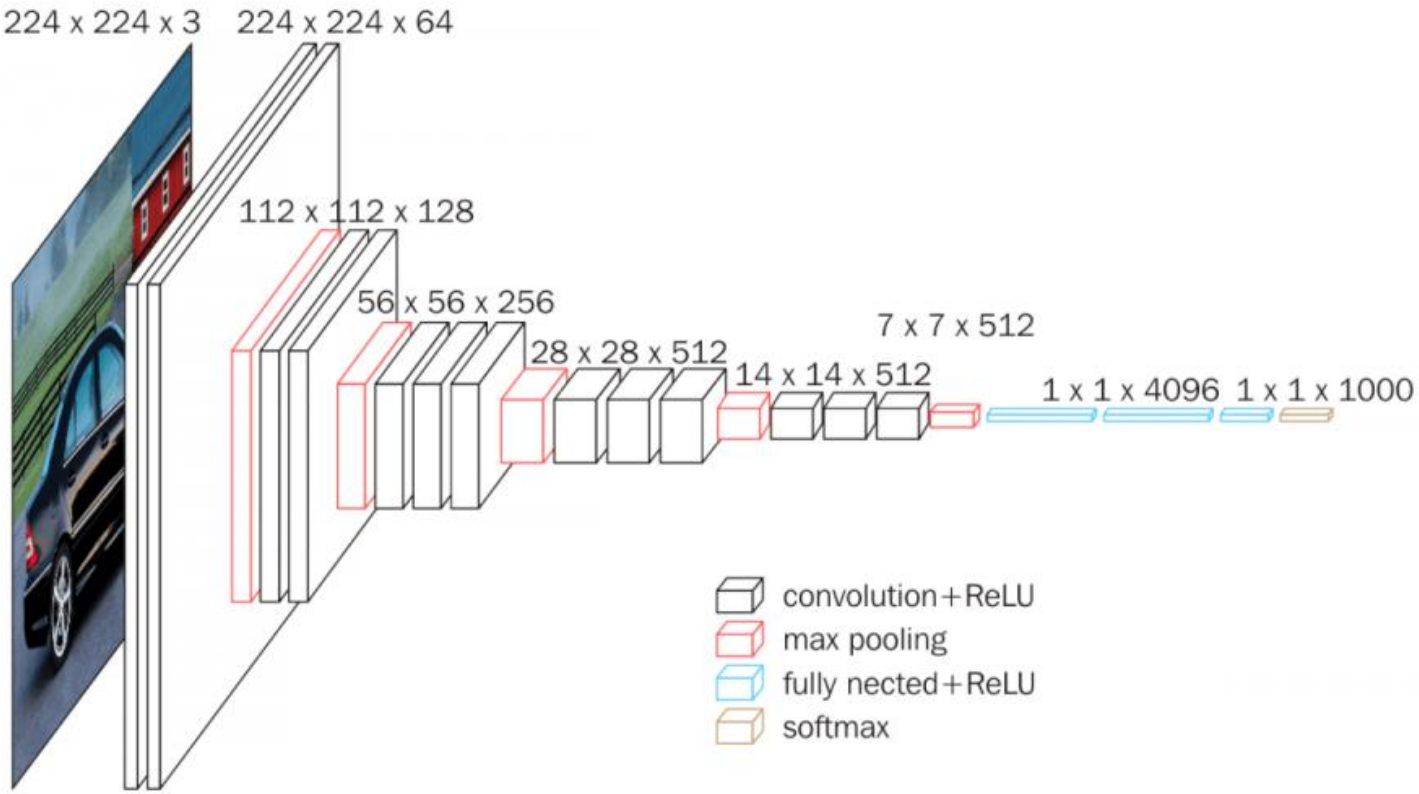
개요

1. CNN은 이미지, 비디오 분석에서 좋은 성능을 보여왔고 관련 연구들이 많이 존재함
2. 관련 연구들의 내용을 살펴보면,
   - Smaller Stride, receptive window size
   - the networks densely over the whole image and over multiple scales
3. VGG에서는 depths 측면에서, small filter인 3x3 convolution filters 만을 이용해서 안정적으로 층을 깊게 쌓는 것을 다룸
4. 결과적으로, 많은 대회에서 상위권 점수를 받았음

1 VGG

ConvNet 구성

모델의 아키텍처



1 VGG

모델 아키텍처 – 전처리

```
VGG_MEAN = [123.68, 116.78, 103.94]    # This is R-G-B for Imagenet

image = tf.random_crop(image, [224, 224, 3])
means = tf.reshape(tf.constant(VGG_MEAN), [1, 1, 3])
image = image - means
```

1. 인풋 데이터 전체의 Mean RGB를 계산해서 각 픽셀에서 뺌
   - Subtract the mean per channel calculated over all images (e.g. VGG_ILSVRC_16_layers)
   - Subtract by pixel/channel calculated over all images (e.g. CNN_S, also see Caffe's reference network)
2. 해당 전처리가 가지는 장점
   - 계산량에서의 장점 : Gradient가 안정감있게 되기 위해서 Small norm을 가지게 함
   - 데이터셋의 편향 : 평균이 0이 아닌 경우 특정 방향으로 계속해서 Gradient가 진행 될 수 있음
   - 데이터의 정규화 : 입력 변수간의 영향도 차이를 줄여줌 (한 변수에 치우치지 않도록 조절)
3. 평균만 빼주는 이유
   - 정확도 감소 : xi가 모두 가깝다면 편차는 매우 작음. 이 값을 나눠주게 되면 정확도의 감소를 유발
   - 데이터의 특성 : 이미지는 (0, 255)에 분포

1 VGG

모델 아키텍처 – Convolutional layers

1. 3 x 3 convolution filters
   - Small receptive filed : left/right, up/down, center를 가장 작은 크기로 잡아냄
2. 1 x 1 convolution filters
   - linear transformation of the input channels (followed by non-linearity)
3. stride, padding size 1
   - the spatial resolution을 보존
4. Max-pooling size 2 x 3 (stride 2)
5. Dense Layer
   - 3개의 FC layer로 구성
   - 처음 두 개는 각 4096개의 채널로 구성, 마지막은 1000개의 target으로 분류
6. 모든 층에는 ReLU를 사용, 정규화 과정은 포함하지 않음
   - 이유 : 성능면에서는 향상 x, 메모리 소비 및 계산 시간만 향상

1 VGG

모델 아키텍처 – Configurations

ConvNet Configuration					
A	A-LRN	B	C	D	E
11 weight layers	11 weight layers	13 weight layers	16 weight layers	16 weight layers	19 weight layers
input (224 × 224 RGB image)					
conv3-64	conv3-64 **LRN**	conv3-64 **conv3-64**	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64
maxpool					
conv3-128	conv3-128	conv3-128 **conv3-128**	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128
maxpool					
conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256 **conv1-256**	conv3-256 conv3-256 **conv3-256**	conv3-256 conv3-256 conv3-256 **conv3-256**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
FC-4096					
FC-4096					
FC-1000					
soft-max					

Table 2: **Number of parameters** (in millions).

Network	A,A-LRN	B	C	D	E
Number of parameters	133	133	134	138	144

층이 깊음에도 불구하고 파라미터들의 수가 적은 것을 볼 수 있음

1 VGG

모델 아키텍처 – Discussion

1. 7 x 7 -> 3 x 3 convolution filters
   - 3 x 3의 layers가 2개이면 5 x 5의 효과를 냄
   - 3 x 3의 layers가 3개이면 7 x 7의 효과를 냄
   - 층을 여러개 쌓으면서 ReLU를 추가하면, 결정 함수가 보다 차별적임 (Discriminative) – 비선형성
   - 파라미터의 수가 더 적음 (3x3 3개 vs 7x7) : 3*(3*3*C*C) < (7*7*C*C)
     - kernel * kernel * in_channel * out_channel
2. 1 x 1 convolution filters
   - convolution의 receptive fields 없이 의사 결정 함수의 비선형성을 증가
   - 계산량 감소

1 VGG

결론

논문에서는 19개의 weight layers를 쌓으면서 깊은 convolutional networks에 대해서 실험을 했음
- 이것은 모델을 깊게 쌓는 것이 정확도에 도움을 준다는 것을 의미
- 3x3의 convolutions를 단순하게 깊게 쌓는 것만으로도 좋은 성능을 낼 수 있음

Method	VOC-2007 (mean AP)	VOC-2012 (mean AP)	Caltech-101 (mean class recall)	Caltech-256 (mean class recall)
Zeiler & Fergus (Zeiler & Fergus, 2013)	-	79.0	86.5 ± 0.5	74.2 ± 0.3
Chatfield et al. (Chatfield et al., 2014)	82.4	83.2	88.4 ± 0.6	77.6 ± 0.1
He et al. (He et al., 2014)	82.4	-	**93.4 ± 0.5**	-
Wei et al. (Wei et al., 2014)	81.5 (85.2*)	81.7 (**90.3***)	-	-
VGG Net-D (16 layers)	89.3	89.0	91.8 ± 1.0	85.0 ± 0.2
VGG Net-E (19 layers)	89.3	89.0	92.3 ± 0.5	85.1 ± 0.3
VGG Net-D & Net-E	**89.7**	**89.3**	92.7 ± 0.5	**86.2 ± 0.3**

Table 9: **Localisation error**

smallest image side		top-5 localisation error (%)	
train (S)	test (Q)	val.	test.
256	256	29.5	-
384	384	28.2	26.7
384	352,384	27.5	-
fusion: 256/256 and 384/352,384		**26.9**	**25.3**

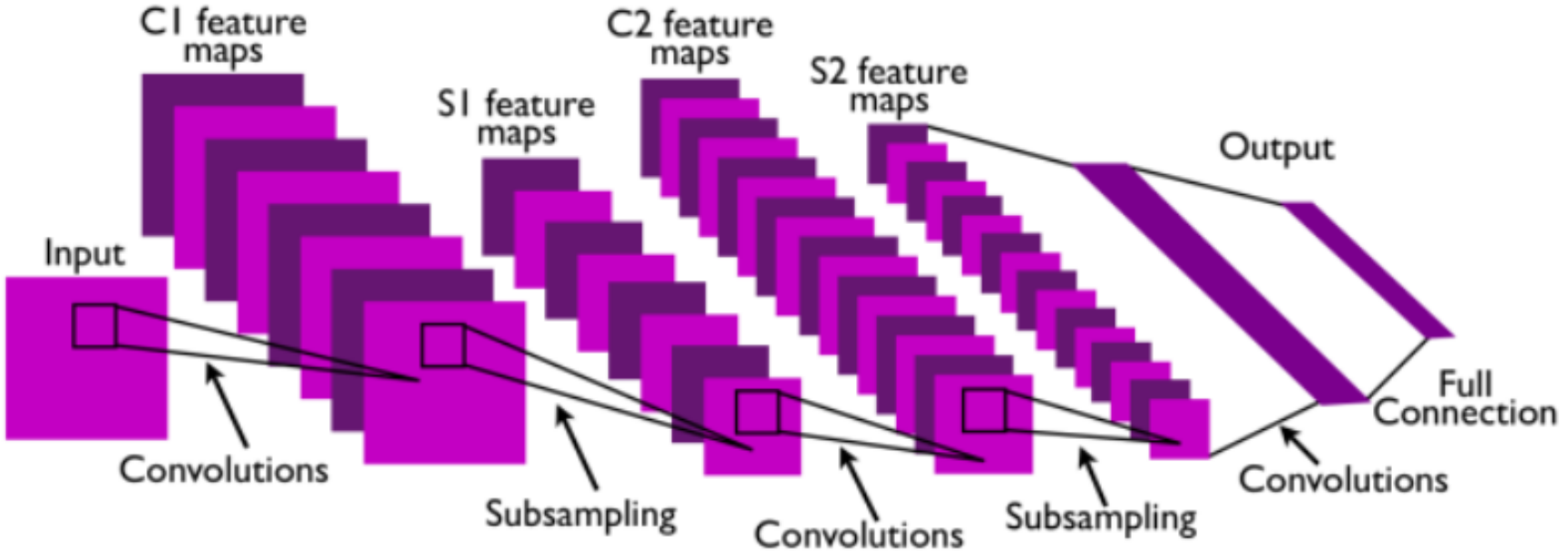
02 Inception-v3

1 CNN (Alexnet)

정의

CNN은 이미지가 들어오면 3가지의 과정을 통해서 Output을 생성합니다.

1. Convolutions : Feature maps 생성
2. Subsampling : 생성된 Feature maps의 차원을 감소
3. Full Connection : Output(Fully Connected Layer)을 산출.



03 Resnet

4 참고 자료

[CNN]

1. https://medium.com/@seoilgun/cnn%EC%9D%98-stationarity%EC%99%80-locality-610166700979
2. https://www.slideshare.net/agdatalab/deep-learning-convolutional-neural-network
3. http://aikorea.org/cs231n/convolutional-networks/
4. https://www.researchgate.net/publication/326816043_FAWCA_A_Flexible-greedy_Approach_to_find_Well-tuned_CNN_Architecture_for_Image_Recognition_Problem
5. https://www.researchgate.net/figure/Model-architectures-for-the-MNIST-and-CIFAR-10-models_tbl10_324558570
6. https://ratsgo.github.io/deep%20learning/2017/04/05/CNNbackprop/
7. https://taewan.kim/post/cnn/
8. https://zzsza.github.io/data/2018/05/14/cs231n-cnn/
9. https://hwiyong.tistory.com/45

4 참고 자료

[Gradient Vanishing & Exploding]

1. https://ayearofai.com/rohan-4-the-vanishing-gradient-problem-ec68f76ffb9b
2. http://keunwoochoi.blogspot.com/2018/01/gradients-explode-deep-networks-are.html

[Jaccobian]

1. http://t-robotics.blogspot.com/2013/12/jacobian.html#.X0OwjcgzaUk
2. https://angeloyeo.github.io/2020/07/24/Jacobian.html

감사합니다