Data Preprocessing & Augmentation

Overview

Data Preprocessing & Augmentation

같은 이미지 데이터라고 하더라도
이를 잘 고쳐서 넣는 것 만으로도 괜찮은 성능 향상을 이루어 낼 수 있다

Table 3: **ConvNet performance at a single test scale.**

ConvNet config. (Table 1)	smallest image side		top-1 val. error (%)	top-5 val. error (%)
	train (S)	test (Q)		
A	256	256	29.6	10.4
A-LRN	256	256	29.7	10.5
B	256	256	28.7	9.9
C	256	256	28.1	9.4
	384	384	28.1	9.3
	[256;512]	384	27.3	8.8
D	256	256	27.0	8.8
	384	384	26.8	8.7
	[256;512]	384	25.6	8.1
E	256	256	27.3	9.0
	384	384	26.9	8.7
	[256;512]	384	**25.5**	**8.0**

같은 이미지를 256px로 변환하냐, 512px로 변환하냐에 따라
정확도의 차이가 달라진다

Very Deep Convolutional Networks for Large-Scale Image Recognition
Simonyan and Zisserman, 2014. https://goo.gl/mbKWhq

Data Preprocessing & Augmengation. Example

사진을 뒤집기
반대쪽을 바라보고 있는 사물(ex: 거울 반사)에 대한 인식률이 올라간다

Data Preprocessing & Augmengation. Example

사진의 일부분을 자르기
일부분이 누락된 사진을 인식할 확률이 올라간다

Data Preprocessing & Augmengation. Example

사진의 색깔을 어둡게/밝게 하기
어두운/밝은 위치에 있는 물체에 대한 인식률이 올라갈 것

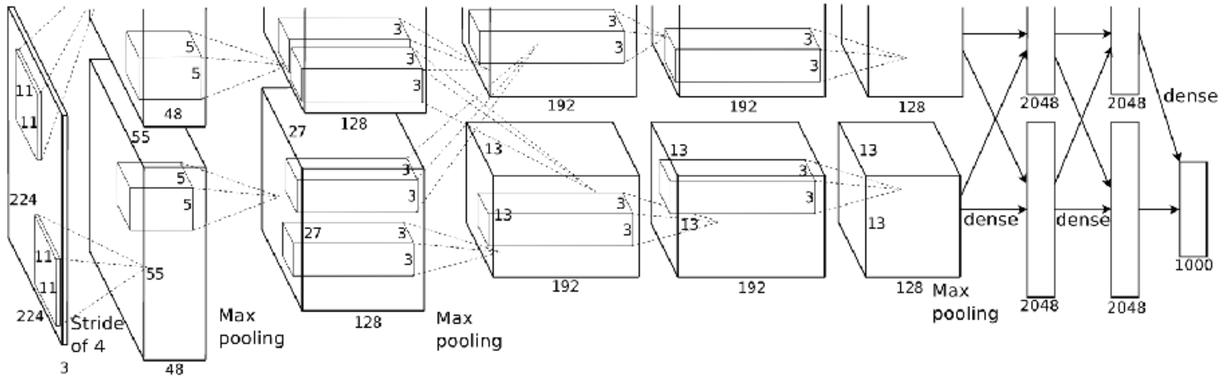
Case Study

AlexNet

가장 기본적인 Data Preprocessing & Augmentation을 제안
사진의 좌우를 반전하거나, 이미지를 랜덤하게 잘라서 사용하는 등의 처리를 했다

Configuration

- **좌우 반전하기**
- **학습할 때는 이미지를 크게 변환한 뒤(256 x 256px), 랜덤하게 224 x 224px을 잘라서 사용**. 이렇게 해서 데이터를 2048배로 불림
- 테스트 할 때는 **이미지를 크게 변환한 뒤(256 x 256px) 이미지에서 좌상/우상/좌하/우하/중간을 224 x 224px만큼 자른 뒤, 좌우반전해서 총 10개의 데이터를 각각 예측하고 그 평균을 구함**.
- PCA를 통해 RGB Channel을 조정해주는 방식을 사용했는데, 요즘은 잘 쓰이지 않는다.



ImageNet Classification with Deep Convolutional Neural Networks
Krizhevsky et al, 2012. https://goo.gl/hqAxZD

VGGNet

가장 심도있는 Data Preprocessing & Augmentation을 시도하였다
시도한 모든 결과를 벤치마킹 후 가장 좋은 설정을 제시함

Configuration

- RGB 값의 평균을 구해서 각각 뺀다. (이러면 loss의 수렴이 빨라진다)
- 이미지 사이즈를 조정할 때, AlexNet과 다르게 256 x 256px만으로 하지 않고, 384 x 384px 512 x 512px로도 한다. (이후 224 x 224px로 랜덤 crop)
- 학습할 때는 모든 이미지마다 랜덤하게 좌우 반전. 테스트때는 좌우 반전 후 두 결과의 평균을 낸다.
- 학습할 때 256px과 512px를 다 넣고, 테스트할 때도 256px, 384px, 512px를 다 넣어서 평균을 내는 것이 가장 낮았다. (7.5%)
- 결과적으로는 아무런 Augmentation을 하지 않았을 때에 비해(10.4%)에 비해 최종적으로는 3% 가까이 에러가 줄어들었다. (7.1%)

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					

Very Deep Convolutional Networks for Large-Scale Image Recognition
Simonyan and Zisserman, 2014. https://goo.gl/mbKWhq

VGGNet

가장 심도있는 Data Preprocessing & Augmentation을 시도하였다
시도한 모든 결과를 벤치마킹 후 가장 좋은 설정을 제시함

Table 3: **ConvNet performance at a single test scale.**

ConvNet config. (Table 1)	smallest image side		top-1 val. error (%)	top-5 val. error (%)
	train (S)	test (Q)		
A	256	256	29.6	10.4
A-LRN	256	256	29.7	10.5
B	256	256	28.7	9.9
C	256	256	28.1	9.4
	384	384	28.1	9.3
	[256;512]	384	27.3	8.8
D	256	256	27.0	8.8
	384	384	26.8	8.7
	[256;512]	384	25.6	8.1
E	256	256	27.3	9.0
	384	384	26.9	8.7
	[256;512]	384	**25.5**	**8.0**

Single Scale Evaluation 결과
데이터를 크게 넣을수록, 그리고 사이즈를 다양하게 넣을수록 정확도가 더 올라간다.

Very Deep Convolutional Networks for Large-Scale Image Recognition
Simonyan and Zisserman, 2014. https://goo.gl/mbKWhq

VGGNet

가장 심도있는 Data Preprocessing & Augmentation을 시도하였다
시도한 모든 결과를 벤치마킹 후 가장 좋은 설정을 제시함

Table 4: **ConvNet performance at multiple test scales.**

ConvNet config. (Table 1)	smallest image side		top-1 val. error (%)	top-5 val. error (%)
	train (S)	test (Q)		
B	256	224,256,288	28.2	9.6
C	256	224,256,288	27.7	9.2
	384	352,384,416	27.8	9.2
	[256; 512]	256,384,512	26.3	8.2
D	256	224,256,288	26.6	8.6
	384	352,384,416	26.5	8.6
	[256; 512]	256,384,512	**24.8**	**7.5**
E	256	224,256,288	26.9	8.7
	384	352,384,416	26.7	8.6
	[256; 512]	256,384,512	**24.8**	**7.5**

Multi Scale Evaluation 결과
Test를 할 때도 이미지의 사이즈를 다양하게 넣는 것이 더 좋은 결과를 내었다.

Very Deep Convolutional Networks for Large-Scale Image Recognition
Simonyan and Zisserman, 2014. https://goo.gl/mbKWhq

VGGNet

가장 심도있는 Data Preprocessing & Augmentation을 시도하였다
시도한 모든 결과를 벤치마킹 후 가장 좋은 설정을 제시함

Table 5: **ConvNet evaluation techniques comparison.** In all experiments the training scale S was sampled from $[256; 512]$, and three test scales Q were considered: $\{256, 384, 512\}$.

ConvNet config. (Table 1)	Evaluation method	top-1 val. error (%)	top-5 val. error (%)
D	dense	24.8	7.5
	multi-crop	24.6	7.5
	multi-crop & dense	**24.4**	**7.2**
E	dense	24.8	7.5
	multi-crop	24.6	7.4
	multi-crop & dense	**24.4**	**7.1**

Multi-crop Evaluation 결과.
Dense와 multi-crop을 동시에 사용하면 약간의 성능 향상이 있었다.

Very Deep Convolutional Networks for Large-Scale Image Recognition
Simonyan and Zisserman, 2014. https://goo.gl/mbKWhq

Others

ResNet은 VGGNet과 거의 동일하므로 패스
그 외에는 rotation을 하거나 rescaling을 하는 등의 Augmentation도 어느정도 효과가 있다

- **rotation**: 0° 에서 360° 사이로 랜덤하게 돌린다.
- **shifting**: 랜덤하게 10픽셀씩 상하좌우로 움직인다.
- **rescaling**: 랜덤하게 1.0 ~ 1.6배로 사진을 키운다.
- **flipping**: 좌우로(내지는 상하로) 사진을 뒤집는다.
- **shearing**: -20° 에서 20° 도 사이로 랜덤하게 찌그러뜨린다.
- **stretching**: 랜덤하게 사진을 1.0 ~ 1.3배로 늘린다.

Summary

이미지를 그대로 넣는 것 보다, 수정을 거치거나 데이터를 강제로 늘린 뒤 넣는 것이 더 좋은 성능을 보장한다.

하지만 성능이 비약적으로 오르지 않는다는 점은 유의할 것. (+3%?) 이 성능 향상이 중요하다고 생각하면 하고, 아니면 안 하는 것이 좋다.

가장 자주 사용하는 것은

- 학습 데이터의 RGB 채널 값의 평균을 빼는 것. (수렴 속도가 빨라진다)
- 좌우 반전
- 이미지를 크게, 다양한 사이즈로 넣기

이 세 가지를 가장 많이 사용한다고 보면 된다.

실습