Softmax Loss를 이용한 Deep Hashing 모델에 대한 연구

이기찬*, 김광수**
*성균관대학교 컴퓨터공학과
**성균관대학교 소프트웨어융합대학
*gc0216@skku.edu **kim.kwangsu@skku.edu

A Study on Deep Hashing Model Using Softmax

Ki-Chan Lee* Kwang-Su Kim**
*Dept. of Computer Engineering, SungKyunKwan University
**College of Software, SungKyunKwan University

요 약

일반적으로 얼굴인식 시스템은 영상에서 추출한 Feature와 DB 상의 Feature를 비교하는 구조를 가지고 있다. 하지만 원하는 Class의 Feature만 보고 DB 상에서 일치하는 Class의 위치를 특정하는 것은 불가능하기에 DB 상의 모든 Feature와 비교하는 절차가 필요하다. DB 크기가 커짐에 따라 처리시간과 메모리상의 문제가 발생하는데, 이 논문에서는 이를 해결하기 위한 Deep Hashing 모델을 제안한다. Softmax 기반의 Loss를 이용하여 학습하였고, 8-bits의 해시를 추출하였을 때 53%의 Feature 일치율을 보였으며, 이를 사용할 경우 DB 평균 대조군을 23% 이하로 줄이는 효과를 볼 수 있을 것으로 추정한다.

1. 서론

일반적으로 얼굴인식 시스템은 먼저 Object Detection 기술을 이용하여 안면 영상을 검출해 내고, 해당하는 안면 영상의 Feature를 추출해 내어 DB 상의 Feature 들과 일대일 비교하여 일정 수치 이상 유사한 데이터를 찾아내는 절차를 가지고 있다. 여기 문제점이 있는데, 추출해낸 Feature를 이용하여 DB를 조회할 때 Hash Rank를 사용한다는 것이다. Hash Rank 방식은 DB 상의 모든 레코드의 Hash와 비교해야 하는 방식으로 최악의 경우, 한 번의 얼굴인식을 위해서 DB 상의 모든 Feature 들과 비교하는 작업이 필요하다. Feature들을 비교하는 작업이 상대적으로 처리시간이 매우 짧기 때문에 DB 크기가 작다면 문제 되지 않는다. 하지만 DB 크기가 커지게 되면 두 가지 문제가 발생하게 된다.

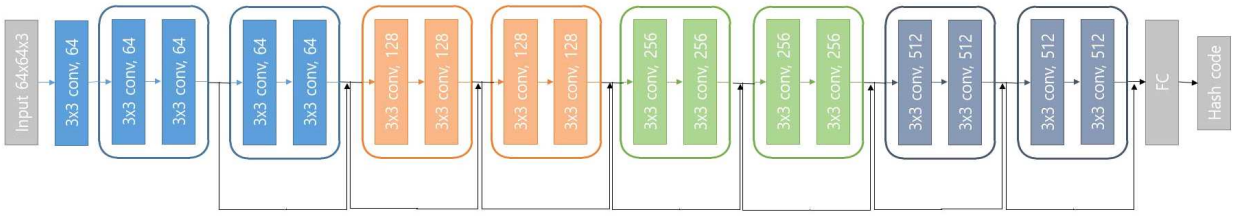
첫 번째로, 처리 속도의 문제이다. 대조해야 할 Feature들의 수가 많아짐에 따라서 안면인식에 필요한 시간이 선형적으로 증가하게 된다.

두 번째로, 메모리상의 문제이다. 네오 페이스[1]의 경우 안면 영상 하나당 Feature의 크기는 5KB이다. DB 상에 존재하는 레코드의 수가 100만 개라고 가정하면 총 DB의 크기는 4.76GB에 달한다. 이와 같은 양의 데이터를 안면인식을 할 때마다 디스크에서 불러온다고 한다면 적어도 매번 수초의 I/O가 발생하게 된다. 따라서 DB 상의 모든 Feature들은 항시 메모리에 로드되어 있어야 하며, 그에 따라서 시스템에 필요한 메모리의 양이 DB 크기에 비례해 증가하게 된다. 위의 두 가지 문제 때문에, 얼굴인식 시스템의 DB를 거대한 DB로 확장하는데 큰 걸림돌이 되고 있다.

이 논문에서는 이러한 문제들을 Deep Hashing을 이용하여 해결해 보고자 한다. 안면 영상의 Semantic 특징을 이해하는 Deep Hashing 기술을 얼굴인식 기술에 보조적으로 적용하여 인덱스를 적용시킬 경우 비교해야 할 feature 후보군을 대폭 줄일 수 있어 위의 언급된 문제들을 해결할 수 있을 것이다.

2. 관련연구

급격한 AI의 발전은 얼굴인식 기술에도 엄청난 발전을 불러왔다. Metric Learning에도 수년간 큰 진전이 있어 Softmax, Euclidean-distance[2], Angular/cosine-margin-based Loss[3] 등 다양한 Metric Learning 기법들이 모색되어 왔다. 이와 같은



(그림 1) Hashing 모델 구조

방법들은 정확도 면에서 아주 우수한 성능을 보여왔다. 하지만 Hash Rank를 이용하기 때문에 확장성에 문제가 있었다.

반면 Distill Hash[4]에서는 Feature를 Hamming Space에 일치시키고, Feature의 비트 수를 줄이는 방향으로 학습시키면서도 우수한 성능을 보였다. 이번 연구에서는 Distill Hash와 같이 Binary Hash Code를 추출하는 딥러닝 모델을 제안하는 것을 목표로 한다.

3. 모델 제안

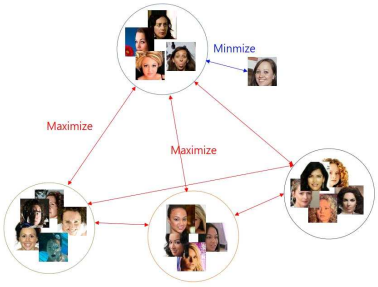
3.1. 모델 구조

딥러닝 모델은 (그림 1)과 같이 Resnet18 모델에 더하여 Full-Connected-Layer로 구성되었다. 마지막 레이어의 경우 tanh 활성화 함수를 이용하여 추출되는 Hash Code 값의 범위를 [-1,1]로 고정하였다.

3.2. 데이터셋

데이터 셋은 celebA 데이터 셋을 사용했다. celebA 데이터 셋을 identity 별로 분류하여 7706개의 Class, 각 Class 당 10장 이상의 데이터를 training, validation set으로 활용, 그 외 997 개의 Class, 각 5장 이상의 데이터를 test set으로 활용하였다.

3.3. 손실함수



(그림 2) Loss 구조

학습 방법은 (그림 2)과 같이 구성된 mini-batch에서 모든 Positive Case(identity가 같은 경우)를 최소화하고 모든 negative case는 최소화하는 방향으로 학습을 진행한다. 학습에 사용한 Loss는 다음과 같다.

$$Loss_{total} = Loss_{softmax} + \alpha \times Loss_{hamming}$$

여기에서 α는 상수이며, $Loss_{softmax}$는 Metric Learning을 위한 Loss, $Loss_{hamming}$는 Feature를 Hamming Space에 근접하도록 하기 위한 Loss이다. $Loss_{softmax}$를 살펴보기 전에 X를 정의하면,

$$X = \begin{pmatrix} X_1 \\ X_2 \\ \cdots \\ \cdots \\ X_{N-1} \\ X_N \end{pmatrix} \quad X_i = \begin{pmatrix} X_{i,1} \\ \cdots \\ X_{i,M} \end{pmatrix} \quad X_{i,j,k} = \begin{cases} 1 \\ -1 \end{cases}$$

$$X_{i,j} = (X_{i,j,1} X_{i,j,1} \ldots\ldots X_{i,j,K-1} X_{i,j,K})$$

X는 딥러닝 모델을 이용하여 추출한 Hash Code이며, N × M × K의 행렬로 N은 Class의 개수, M은 Class의 데이터의 수, K는 Feature 성분의 개수를 의미한다. $Loss_{softmax}$는 다음과 같다.

$$Loss_{softmax} = Softmax(Cosines \times s)$$

$$Softmax(X) = \sum_{i=1}^{N} -\log\left(\frac{\sum e^{X_{i,i}}}{\sum_{j=1}^{N}\sum e^{X_{i,j}}}\right)$$

$$Cosines = \begin{pmatrix} X_1X_1^T \div (L_1L_1^T) & \cdots & \cdots & X_1X_N^T \div (L_1L_N^T) \\ \cdots & & \cdots & \cdots \\ \cdots & & \cdots & \cdots \\ X_NX_1^T \div (L_NL_1^T) & \cdots & \cdots & X_NX_N^T \div (L_NL_N^T) \end{pmatrix}$$

여기에서 s는 상수이며, Cosines는 N × N × M × M 행렬이고 Hash Code 사이의 cosine 각도를 나

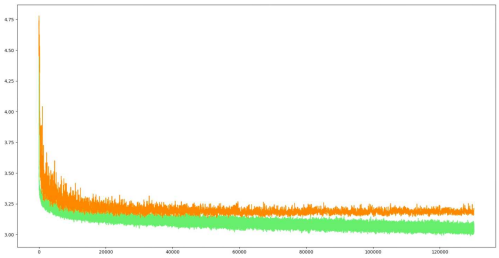
타낸다. Softmax 함수는 기존의 Softmax를 응용한 것이나 기존의 weight 값과 데이터를 내적 한 값이 아닌 Feature와 Feature 사이의 Consine 값을 이용한다.

$Loss_{hamming}$는 다음과 같이 정의된다.

$$Losss_{hamming} = \|1 - \|X\|_1\|_2$$

Feature의 절댓값이 1에 근접하도록 하는 loss이다. weight 값이 0이 되는 경우를 방지하기 위해 L1이 아닌 L2 loss를 이용한다.

4. 구현 및 결과분석



(그림3)학습 러닝 커브

(그림 3)은 해당 모델의 학습 그래프이다. Weight Decay와 Data Augmentation 기법을 이용하여 Validation Loss가 눈에 띄게 증가하는 구간 없이 안정적으로 학습이 진행되는 것을 볼 수 있다.

featur e	Positive-Pair			Negative-Pair		
	equal	1	2	equal	1	2
8-bits	0.536	0.737	0.876	0.042	0.081	0.169
12-bits	0.387	0.602	0.765	0.007	0.016	0.036
16-bits	0.243	0.407	0.560	0.004	0.009	0.018
20-bits	0.192	0.319	0.445	0.003	0.006	0.011
24-bits	0.212	0.331	0.436	0.003	0.005	0.008
28-bits	0.217	0.327	0.422	0.003	0.004	0.007
32-bits	0.132	0.204	0.285	0.002	0.003	0.005

<표1> hash code간 hamming distance 분포

다음 <표 1>은 Class가 일치하는 Positive-pair와 Class가 불일치하는 Negative-pair 사이의 Hamming Distance의 분포 비율을 Feature의 길이 별로 나타낸 것이다. equal은 완전히 일치하는 경우, 1과 2는 각각 데이터 사이의 거리가 1, 2 이하인 경우를 나타낸다. 8비트의 경우 같은 Class에서 53%의 확률로 일치하며 다른 Class에 대해서 4프로의 확률로 일치한다는 것을 알 수 있다.

<표 2>를 보면 8-bits의 Hash Code를 사용하고 Hamming Distance가 2인 Hash Code까지 탐색하는 경우 23% 정도로 대조해야 할 Feature 후보군들의 수를 줄이는 효과가 있을 것으로 추정할 수 있다.

feature	Estimation		
	Equal	1-diff	2-diff
8-bits	0.5055	0.3312	0.2266
12-bits	0.6199	0.4125	0.2596
16-bits	0.7616	0.6016	0.4554
20-bits	0.8118	0.6866	0.5650
24-bits	0.7910	0.6746	0.5722
28-bits	0.7857	0.6776	0.5844
32-bits	0.8695	0.7990	0.7192

<표 2> 예상 비교 후보군 감소 효과

다음 (그림 4)는 8-bits Hash 모델을 사용하였을 때 다른 Class이지만 같은 Hash로 분류된 것을 정리한 것이다. 보는 바와 같이 같은 Hash Code를 가지는 데이터들은 비슷한 특징을 가지고 있는 것을 확인할 수 있다. 즉, Hash Code가 데이터의 Sementic한 특징을 잘 보존하고 있다는 것을 알 수 있다.

(그림 4) 동일 Hash Code 다른 Class

5. 결론

구현 결과 8-bits의 Hash code를 이용하는 경우 최대 23%까지 DB 대조군을 줄일 수 있다는 결과가 나왔다. 그 결과 검색 시간이 3배 이상 향상될 수 있다고 추정할 수 있다. 하지만 이 결과에는 Hash Code를 추출하는 시간은 포함되지 않았는데, 기존의 Face Verification 신경망에 Built-In 시켜 Face Verification을 위한 Feature와 Hash Code를 동시 추출한다면 Hash Code를 추출하는 시간은 무시할 수 있을 것이다. 그 외의 변수로 별도의 탐색 알고리즘 또는 CPU의 캐시 구조 등에서 오는 오버헤드 등의 변수가 있다.

참고문헌

[1] 다인스(주) NeoFace.
http://www.da-ins.co.kr/m31.php

[2] Florian Schroff, Dmitry Kalenichenko, James Philbin; FaceNet: A Unified Embedding for Face Recognition and Clustering, Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) 2015, pp. 815-823, 2015

[3] Hao Wang, Yitong Wang, Zheng Zhou, Xing Ji, Dihong Gong, Jingchao Zhou, Zhifeng Li, Wei Liu; CosFace: Large Margin Cosine Loss for Deep Face Recognition, Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2018, pp. 5265-5274, 2018

[4]Erkun Yang, Tongliang Liu, Cheng Deng, Wei Liu, Dacheng Tao; DistillHash: Unsupervised Deep Hashing by Distilling Data Pairs, Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2019, pp. 2946-2955