

Internship Seminar

# Introduction to Recommendation System

July 26th, 2021  
SK Planet  
Taegyeong, Eo

## 추천 시스템이란?

- ✓ 사용자가 사용하여 만족할만한 아이템을 보여줌
- ✓ 항목이란 상품, 영상 컨텐츠, 문서, 노래 등 다양한 도메인에 적용될 수 있음
- ✓ 유튜브엔 하루에 72만 시간의 비디오가 업로드
- ✓ 방대한 컨텐츠를 적절히 골라주는 것이 플랫폼의 경쟁력이 됨



**70%**  
of **YouTube views**  
are generated by its  
**artificial  
intelligence-based  
algorithm**



**81%** of surveyed US adults  
occasionally or regularly  
watch videos recommended by the  
**YouTube algorithm.**

**Everything is a Recommendation**

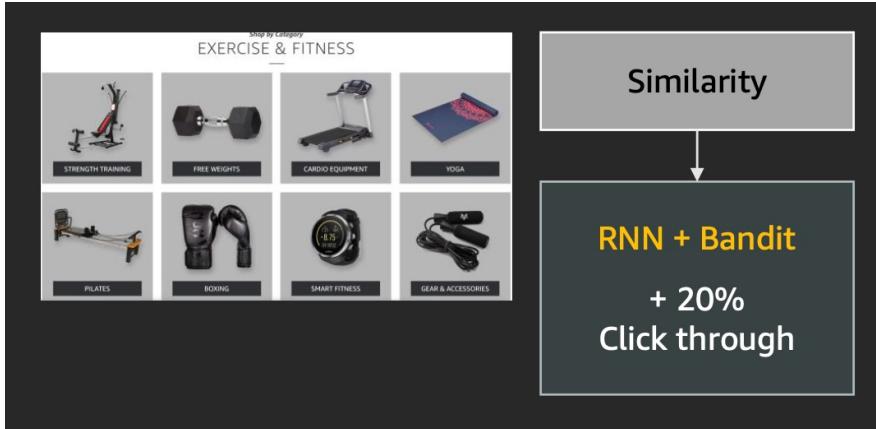


Over 80% of what  
people watch  
comes from our  
recommendations

Recommendations  
are driven by  
**Machine Learning**

# 1 Introduction

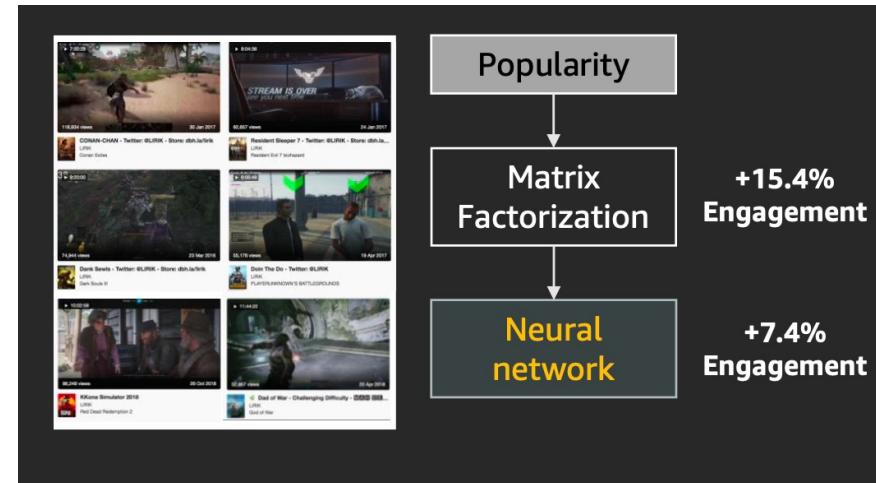
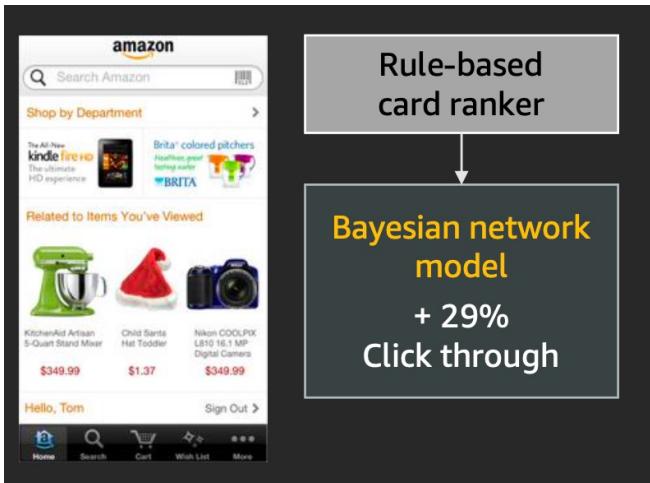
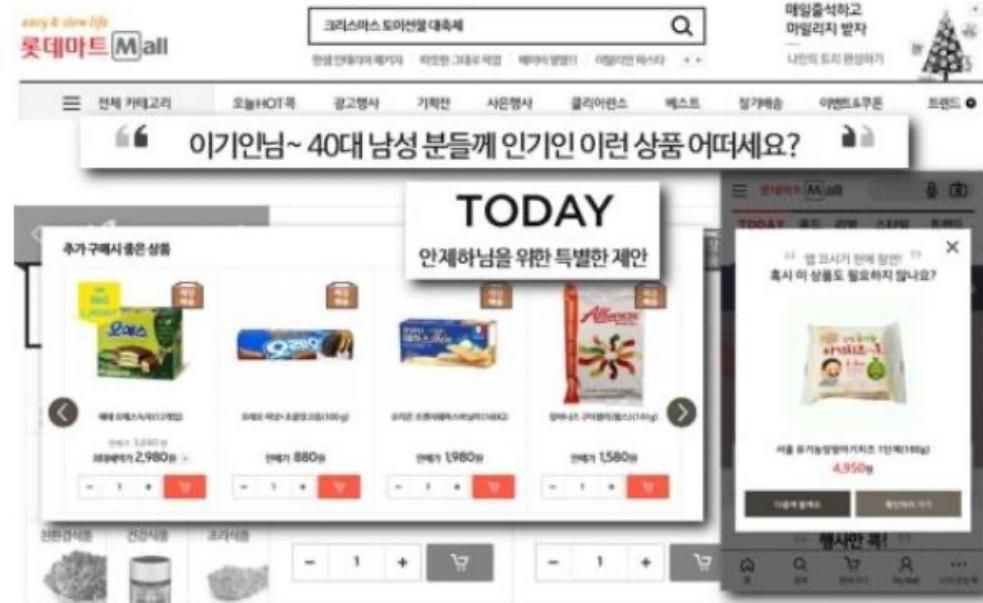
## 적용 사례



### 롯데마트몰, 개인화 추천으로 매출 30% 증가

| 2017.02.10

가- 가+



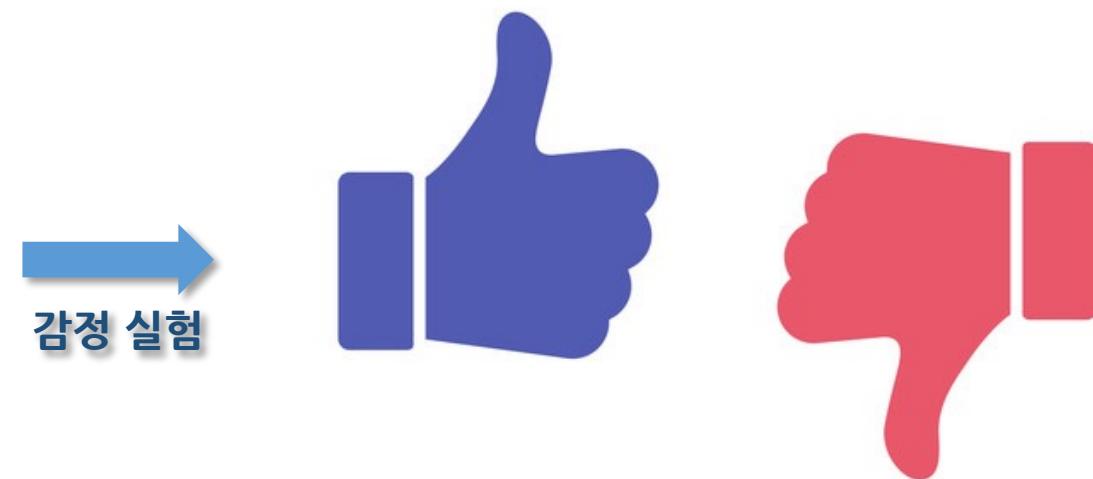
## 1 Introduction

# Facebook 대규모 감정 실험

- ✓ 70만명을 상대로 한 대규모 실험
- ✓ 긍정/부정 게시글을 인위적으로 조절하여 게시 → 긍정 게시물에 긍정적인 반응
- ✓ 삶과 밀접한 관련이 있는 추천 알고리즘



사실일까?



노출 게시물의 감정에 따라  
사용자의 감정이 변화

## 1 Introduction

# 추천 vs 검색

- ✓ **검색이란?** 대용량 문서에서 주어진 질의와 관련된 상위 N개의 문서를 찾아주는 시스템
- ✓ 검색은 사용자가 쿼리를 통해 Active → 의도에 맞게 결과를 내주는 것이 중요한 시스템임
- ✓ 추천은 문서가 아닌 item을 반환, 쿼리가 주어지는 것이 아닌 사용자 정보(session, context)가 주어짐
  - item - 페이스북의 친구, 플레이 스토어의 앱, 유튜브의 비디오, 멜론의 노래, 카카오 선물하기의 상품 등
  - context - 사용자 정보(검색기록, 이전 구매 상품 등), 시간, 기기 정보, 세션 히스토리 등
- ✓ 사용자의 명확한 의도가 주어지지 않기 때문에 훨씬 더 많은 데이터를 고려해야함

Pull model: "people looking for information" ACTIVE

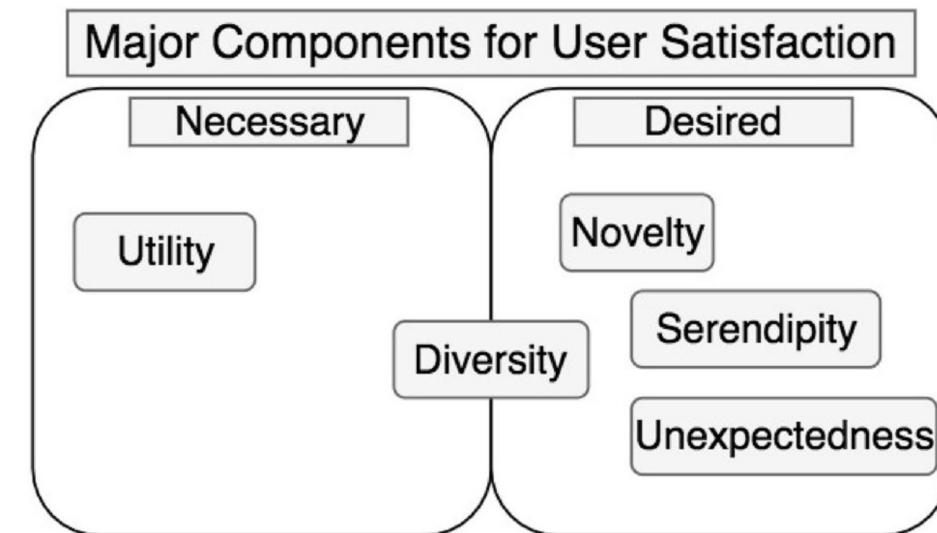
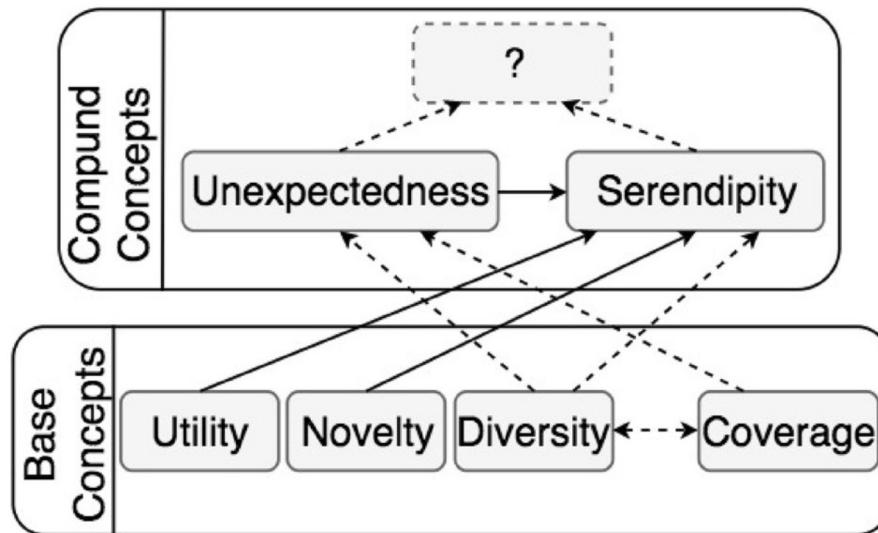


Push model: "information looking for people" PASSIVE



# 추천 시스템의 특성

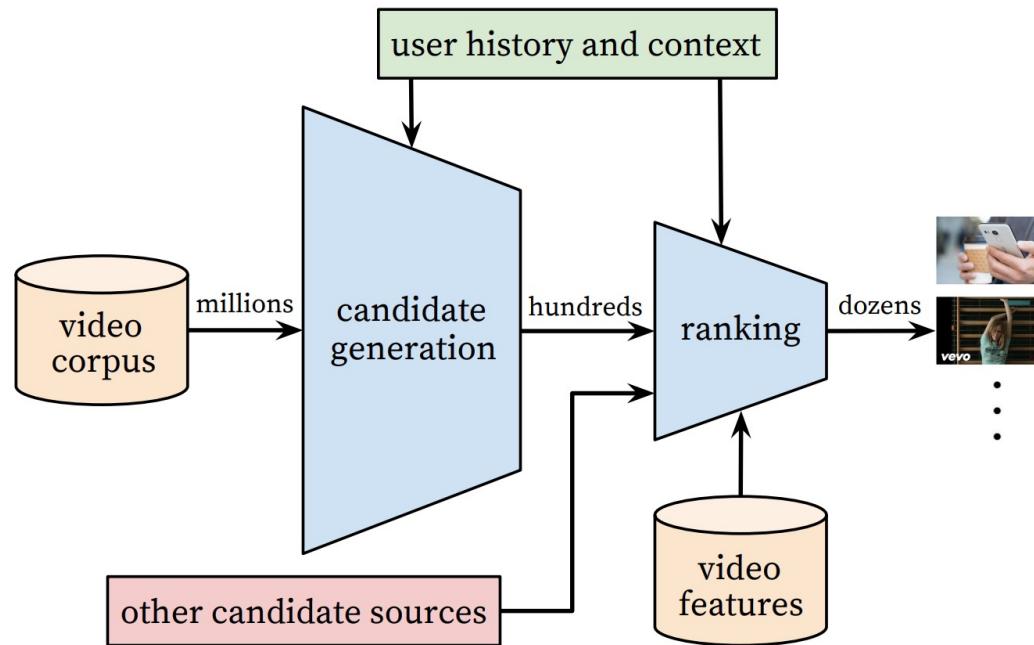
- ✓ **Diversity** - 유저의 히스토리와 가장 유사한 항목만 추천하는 것보다 다양한 카테고리를 추천하는 것이 만족도가 높음
- ✓ **Fairness** - 아이템의 최소 노출 횟수를 보장하여 소외되는 아이템이 없어야 함
- ✓ **Freshness** - 실시간으로 새로운 유저와 아이템이 업데이트 되는데 이에 대응할 수 있어야 함
- ✓ **Robustness** - 악의적으로 시스템의 신뢰성에 영향을 줄 수 있는 feedback에 대처가 가능해야 함
- ✓ **Trust** - 설명 가능한 시스템이여야 사용자에게 신뢰를 줄 수 있음



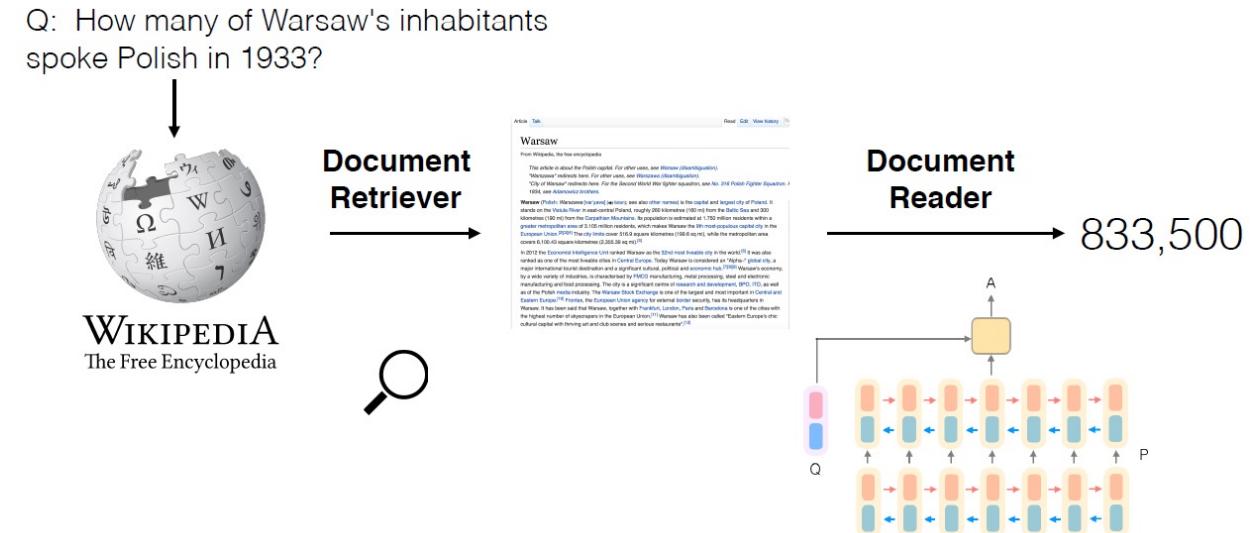
## 2 Architecture

# 추천 시스템의 구성

- ✓ **Candidate Generation** - 컨텐츠가 너무 많기 때문에 러프하게 필터링하는 단계
- ✓ **Ranking** - 필터링된 컨텐츠에 대해서 사용자의 정보를 추가적으로 반영하여 상위 항목 정렬



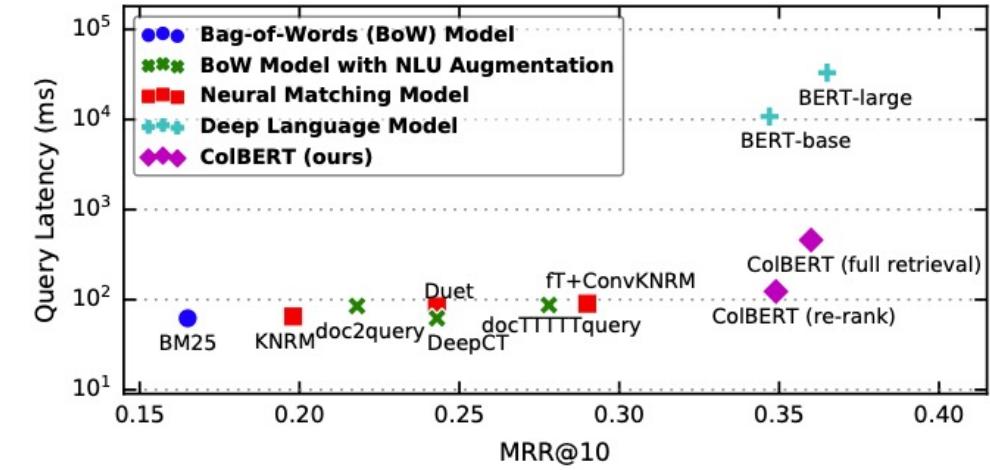
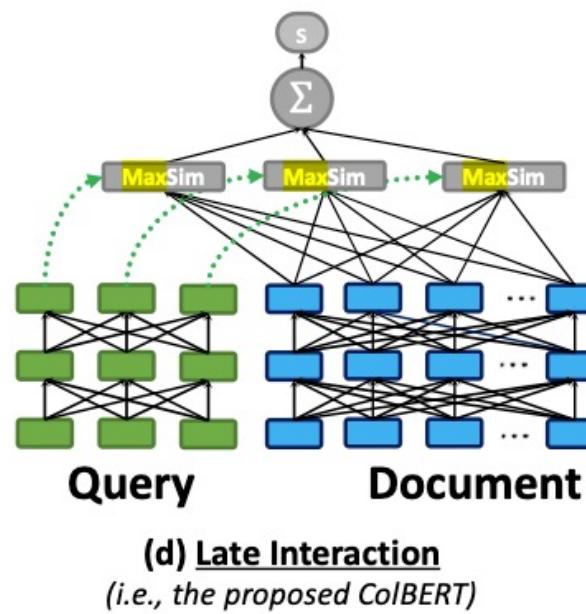
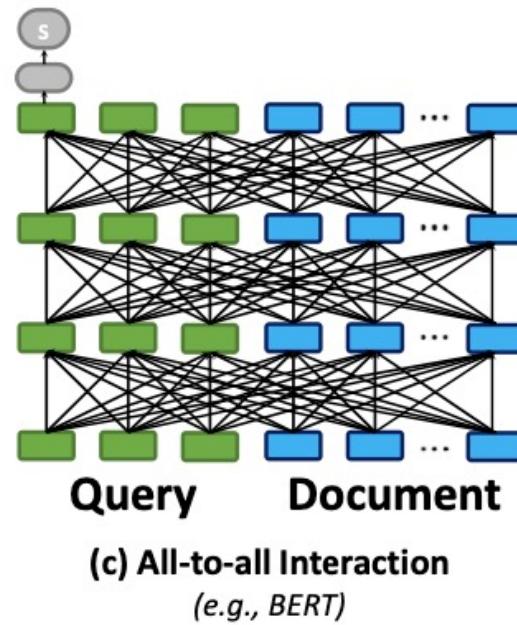
※ Deep Neural Networks for YouTube Recommendations (2016)



※ Reading Wikipedia to Answer Open-Domain Questions (2017)

# IR with ColBERT

- ✓ BERT의 내부 interaction 효율적으로 개선 → 성능은 BERT급, 추론시간은 BM25와 유사
- ✓ Faiss의 IVFPQ index를 사용하여 인덱싱 상용 검색모델로 충분히 사용가능한 수준의 레이턴시
  - Product Quantization
  - K-means Clustering



## 2 Architecture

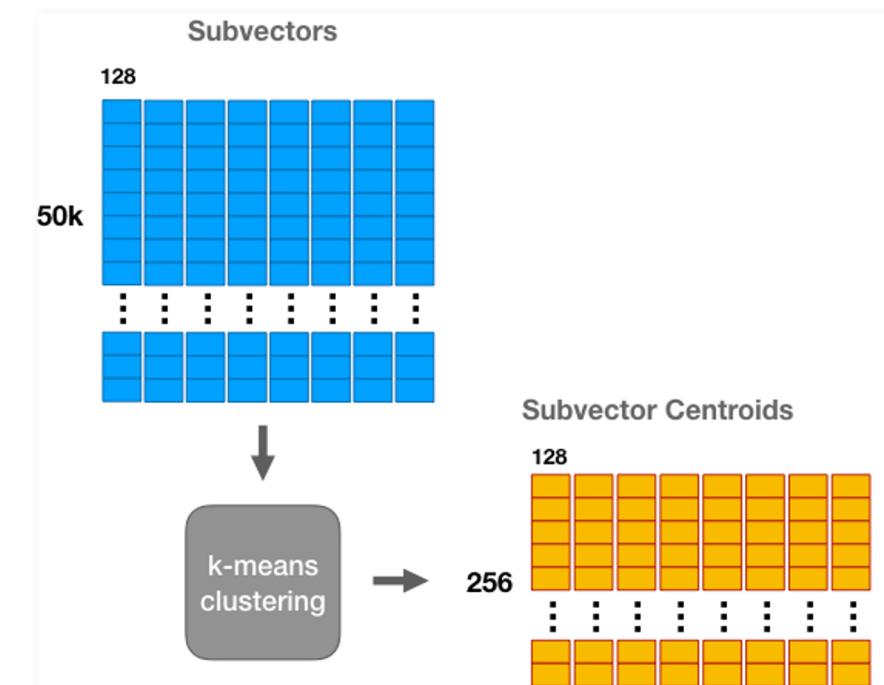
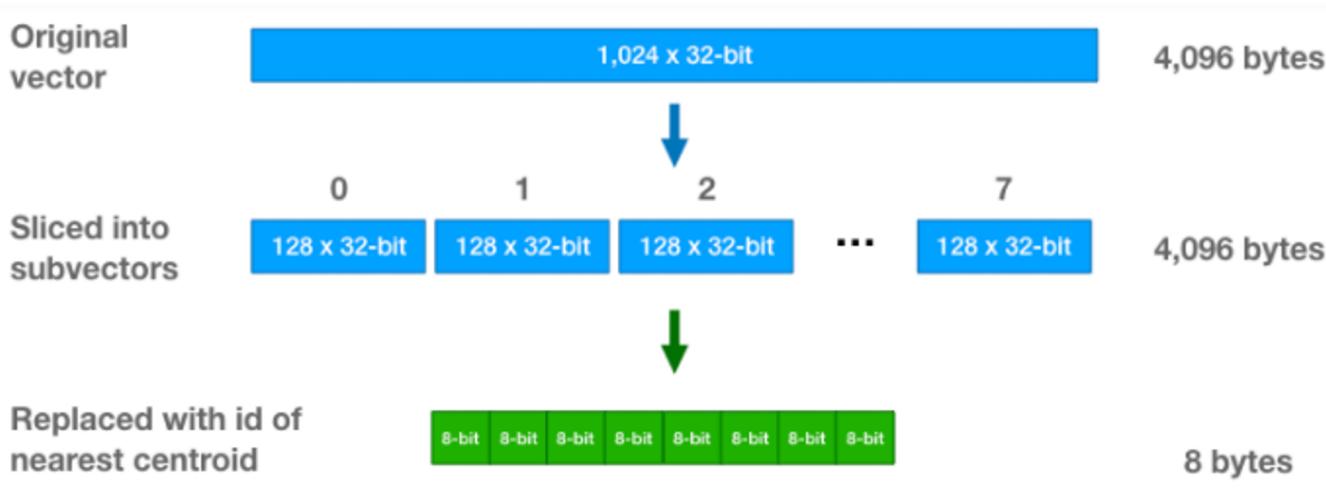
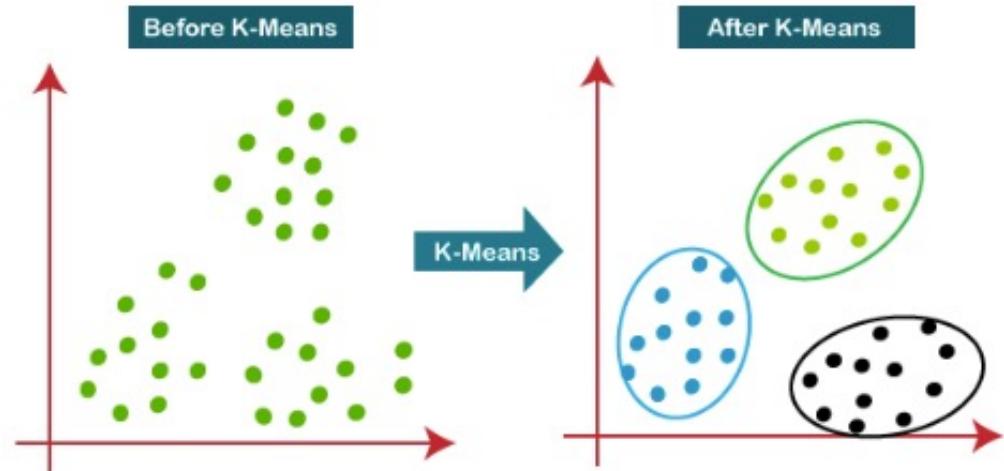
# Indexing Passage

### ✓ K-means Clustering

- 1) 그룹 수 만큼 랜덤으로 샘플링
- 2) 그룹이 할당된 가장 가까운 점의 그룹으로 편입을 반복

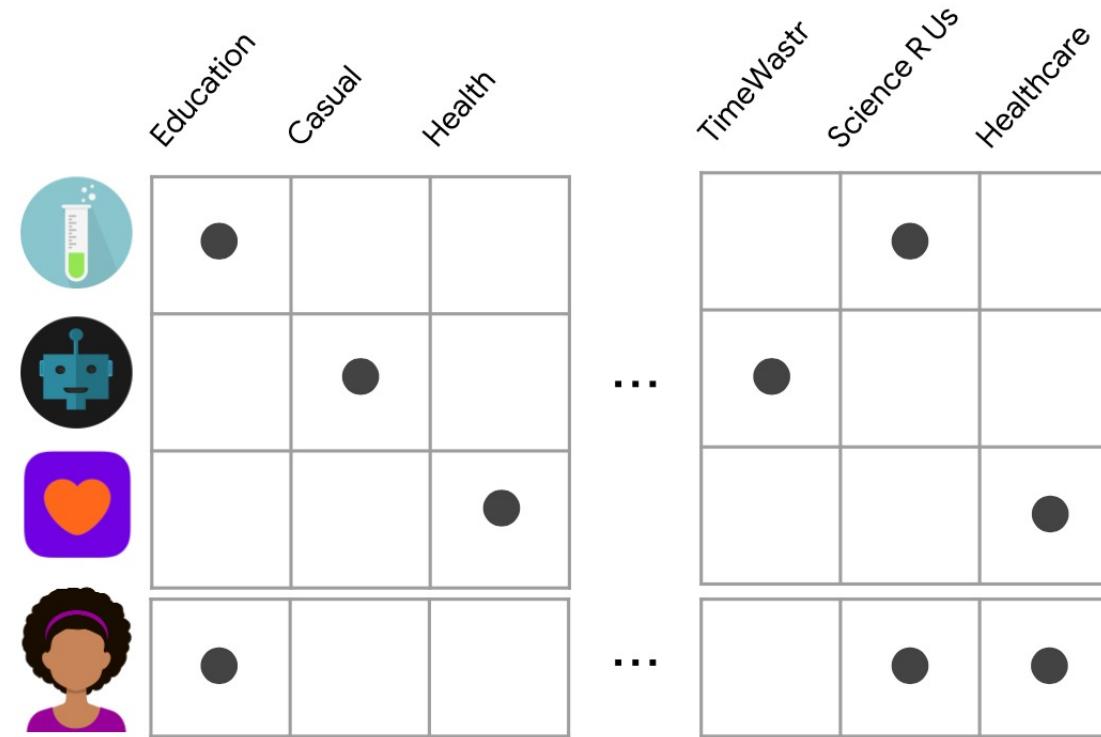
### ✓ Product Quantization

- 1) 하나의 벡터를 8개의 서브 벡터로 분할
- 2) K-means 클러스터링 → 클러스터의 중심점으로 대체하여 압축



# Content-based Filtering

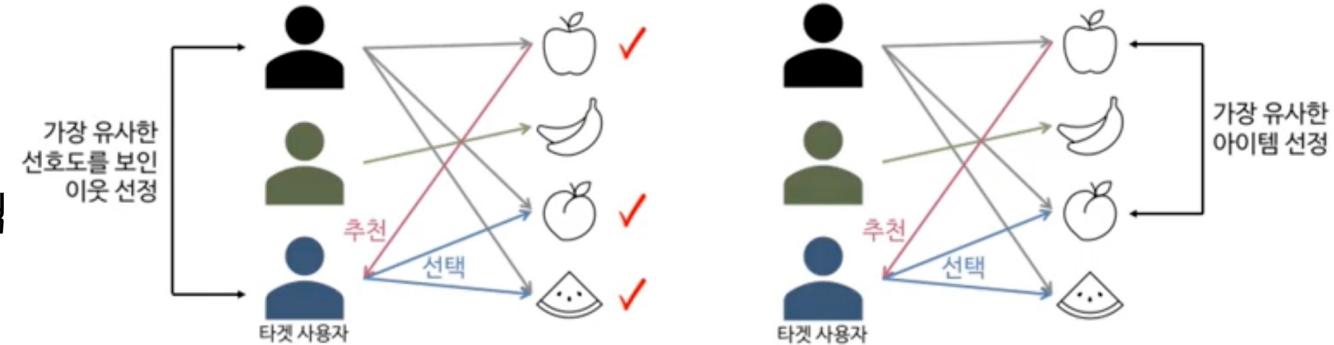
- ✓ 아이템의 정보를 바탕으로 유사한 아이템을 추천
- ✓ 아이템 vector간의 유사도를 측정하여 상위 항목 필터링
- ✓ **장점?** Cold-Start 문제가 발생하지 않음
- ✓ **단점?** Feature engineering에 많은 비용 소모, 사용자에 대한 정보를 반영하기 어려움, Overspecialization



# Collaborative Filtering

- ✓ 사용자와 아이템간의 유사성을 동시에 고려
- ✓ 장점? Feature engineering X, 임베딩만 학습하면 됨
- ✓ 단점? Cold-Start 문제, 다른 정보를 반영하기 어려움

기억기반 모델(사용자, 아이템)



학습기반 모델(Matrix Factorization)



≈

.9	-1	1	1	-.9
-.2	-.8	-1	.9	1
1	.1	.88	-1.08	0.9
-1	0	-0.9	1.0	-1.0
.2	-1	0.38	0.6	1.2
.1	1	-0.11	-0.9	-0.9
			1.0	0.91

# Ranking

- ✓ 러프하게 필터링된 컨텐츠를 대상으로 더 많은 메타정보를 반영하여 추천
- ✓ 유저에 대한 정보(히스토리, 위치 등) + 컨텐츠에 대한 정보(제목, 설명, 카테고리 등)
- ✓ 다음과 같은 특성을 유지하는 것이 중요

## ❖ Freshness

- 최근 정보만을 반영하는 것은 X
- 유저의 히스토리 반영 → 새로운 아이템 추천
  - 실시간 데이터 트레이닝
  - 유저 클러스터링
  - Deep Learning
  - Feature Engineering

## ❖ Diversity

- 유사도를 기반 → 비슷한 아이템만 추천될 수 있음
  - Multi-task Learning

## ❖ Fairness

- 편향되어 학습될 수 있는 요소를 제거
  - **Positional bias**: 자주 노출되는 컨텐츠가 더 많이 클릭되는 경향

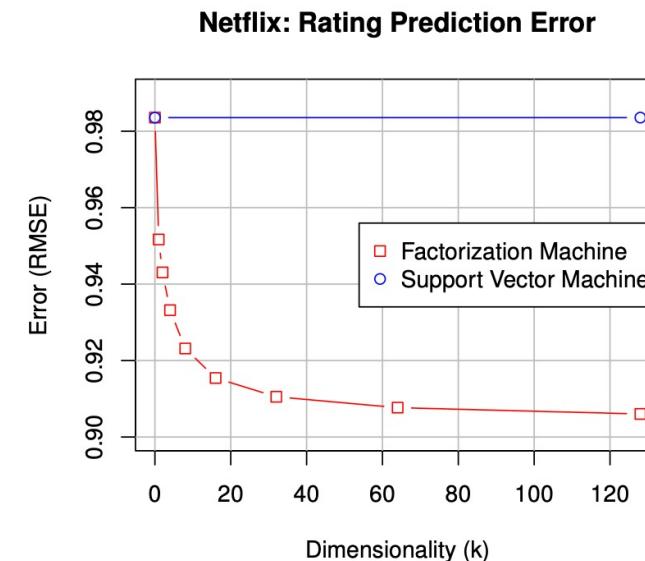
# Factorization Machine

- ✓ 선형모델에서 feature간 interaction을 개선한 모델
  - ✓ 연산을 최적화하여 선형시간에 추론가능

$$\hat{y} = w_0 + \boxed{\sum_{i=1}^n w_i x_i} + \boxed{\sum_{i=1}^n \sum_{j=i+1}^n (\vec{v}_i \cdot \vec{v}_j) x_i x_j}$$

선형 회귀
Feature interaction

- ✓ 선형 모델에 비해 성능향상
  - ✓ Field-aware FM, DeepFM, xDeepFM으로 확장



# FM: 실습

## 데이터 로드

한국영화  
평점 데이터셋  
(KMRD)



- ✓ 유저가 영화에 남긴 평점
- ✓ 영화에 대한 정보
  - 감독, 출연자
  - 제작한 나라
  - 장르

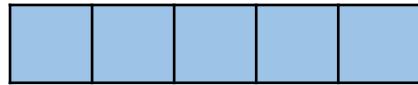
## 데이터 변환

	user_0	user_3	user_4	user_8	user_19	user_25	user_28
137023	0	0	0	0	0	0	0
92868	0	0	0	0	0	0	0
94390	0	0	0	0	0	0	0
22289	0	0	0	0	0	0	0
80155	0	0	0	0	0	0	0

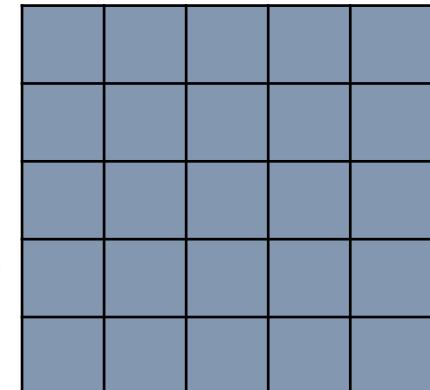
- ✓ One-hot 인코딩
  - 레이블이 여러개인 경우 새로운 컬럼으로 생성
- ✓ 유저 인덱스 + 영화 인덱스 + 영화 정보
  - 변환한 데이터를 Concat
- ✓ 유저가 평가한 점수를 레이블로 둠

## 모델에 적용

W (Linear weight)



V (Interaction weight)



n\_feature

n\_factor

## Field-aware FM

- ✓ FM에서는 필드간의 interaction을 하나의 공간에 학습하여 효율이 떨어짐
- ✓ FFM은 필드의 개수만큼 학습 파라미터를 두어 필드간 interaction을 필드를 구분하여 학습
- ✓ 파라미터 수가 필드의 개수만큼 늘어남에 따라 시간복잡도 증가
- ✓ CTR prediction에서 현업에서도 많이 쓰이고 대회에서도 우수한 성적을 거두는 중

데이터셋

Clicked	Publisher (P)	Advertiser (A)	Gender (G)
Yes	ESPN	Nike	Male

FM

$$\mathbf{w}_{ESPN} \cdot \mathbf{w}_{Nike} + \mathbf{w}_{ESPN} \cdot \mathbf{w}_{Male} + \mathbf{w}_{Nike} \cdot \mathbf{w}_{Male}$$

FFM

$$\mathbf{w}_{ESPN, A} \cdot \mathbf{w}_{Nike, P} + \mathbf{w}_{ESPN, G} \cdot \mathbf{w}_{Male, P} + \mathbf{w}_{Nike, G} \cdot \mathbf{w}_{Male, A}$$

→ 필드별로 파라미터를 따로 학습

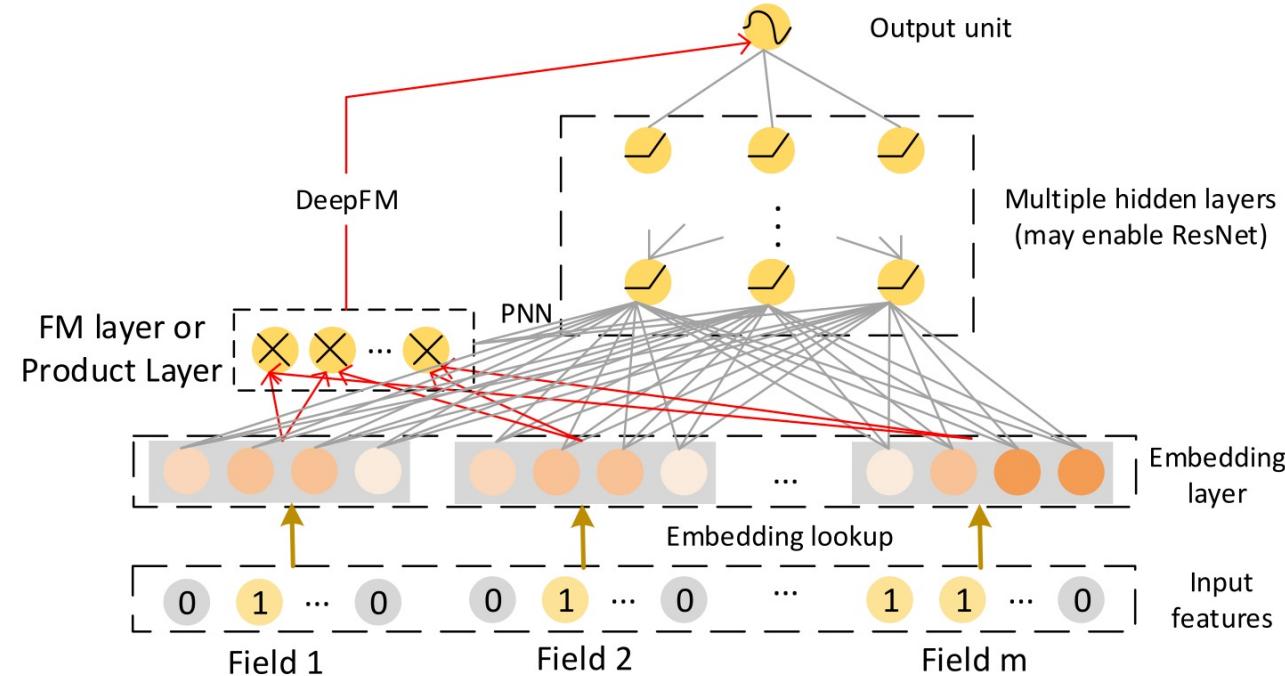
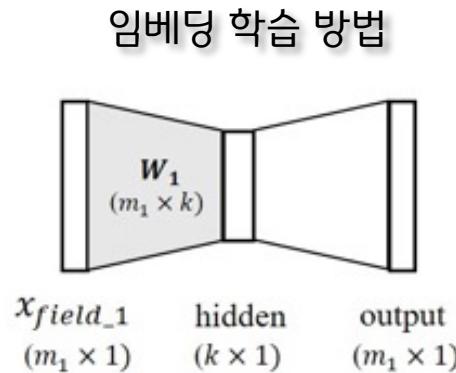
## FFM: 성능평가

- ✓ 대체로 선형모델에 비해 성능향상
- ✓ 특정 데이터셋에서 큰 폭으로 성능향상
- ✓ FM계열에서 시간 비용 대비 성능비가 가장 좋은 모델이라고 생각됨
- ✓ 레이턴시가 중요한 시스템에서는 가장 먼저 고려해 봄야할 알고리즘

Data set	statistics			logloss			
	# instances	# features	# fields	LM	Poly2	FM	FFM
KDD2010-bridge	20,012,499	651,166	9	0.27947	0.2622	0.26372	<u>0.25639</u>
KDD2012	149,639,105	54,686,452	11	0.15069	0.15099	0.15004	<u>0.14906</u>
phishing	11,055	100	30	0.14211	0.11512	<u>0.09229</u>	0.1065
adult	48,842	308	14	0.3097	0.30655	0.30763	<u>0.30565</u>
cod-rna (dummy fields)	331,152	8	8	0.13829	0.12874	<u>0.12580</u>	0.12914
cod-rna (discretization)	331,152	2,296	8	0.16455	0.17576	0.16570	<u>0.14993</u>
ijcnn (dummy fields)	141,691	22	22	0.20093	0.08981	0.07087	<u>0.0692</u>
ijcnn (discretization)	141,691	69,867	22	0.21588	0.24578	0.20223	<u>0.18608</u>

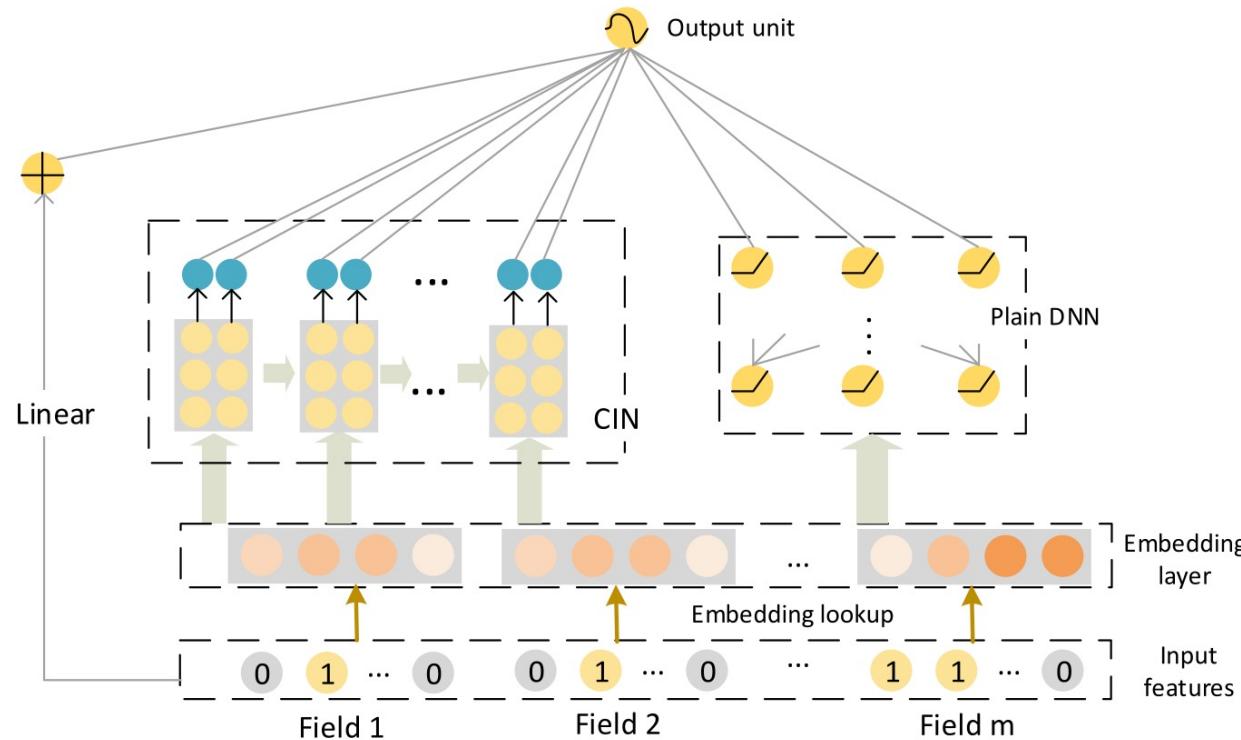
# DeepFM

- ✓ 기존의 FM은 낮은 차원의 interaction을 학습하는데 그쳤음
- ✓ Deep component에선 뉴럴넷을 깊게 쌓아 높은 차원의 feature interaction을 가능하게 함
- ✓ FM component에선 낮은 차원의 interaction을 학습
- ✓ FM Layer pre-training X, 임베딩을 공유 → 효율적



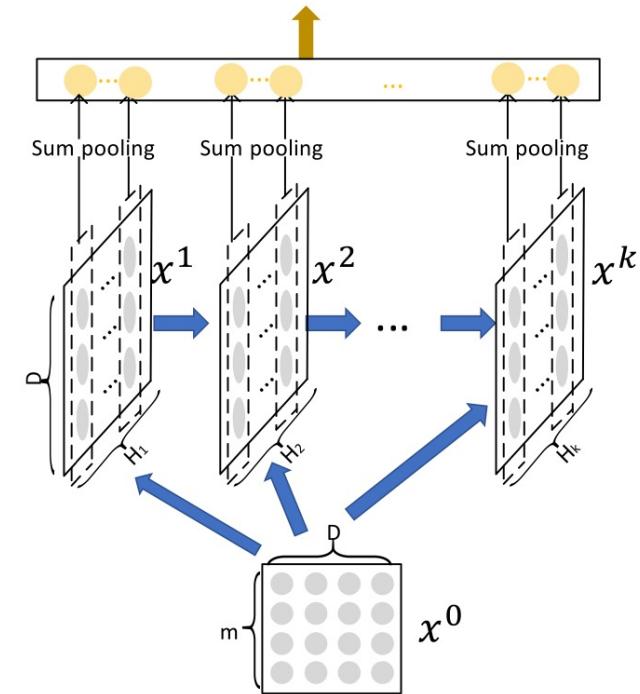
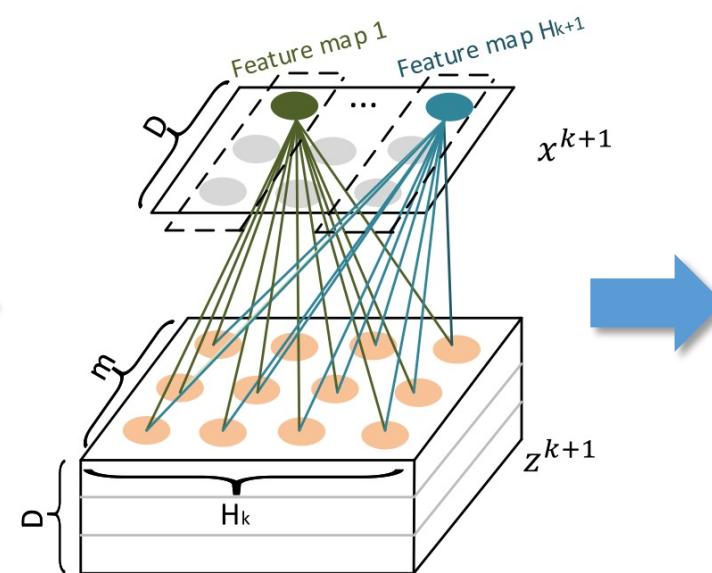
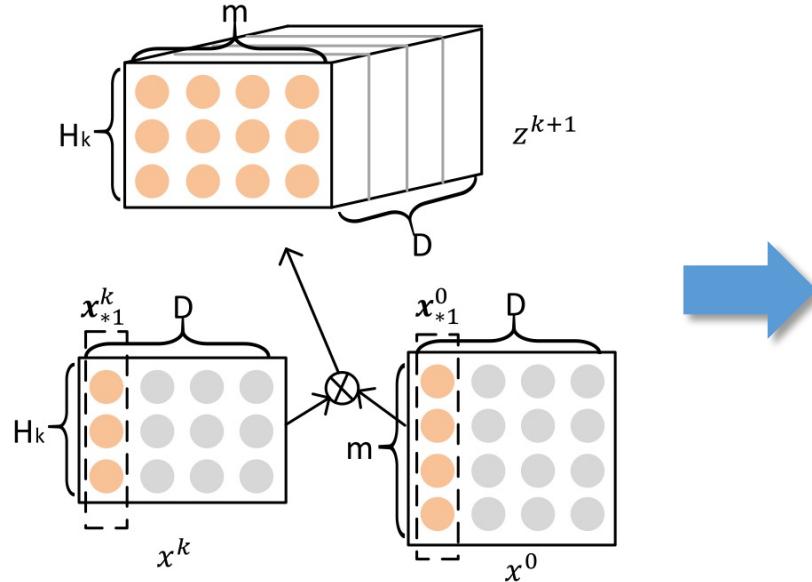
# xDeepFM

- ✓ FM component의 interaction을 vector-wise하게 하고 싶음
- ✓ Compressed Interaction Network(CIN) 도입
- ✓ Deep component에서 point-wise하게 CIN component에서 vector-wise하게 학습하여 결합
- ✓ FM계열에서 가장 좋은 성능



# xDeepFM: Compressed Interaction Network

## Vector-level Interaction 과정



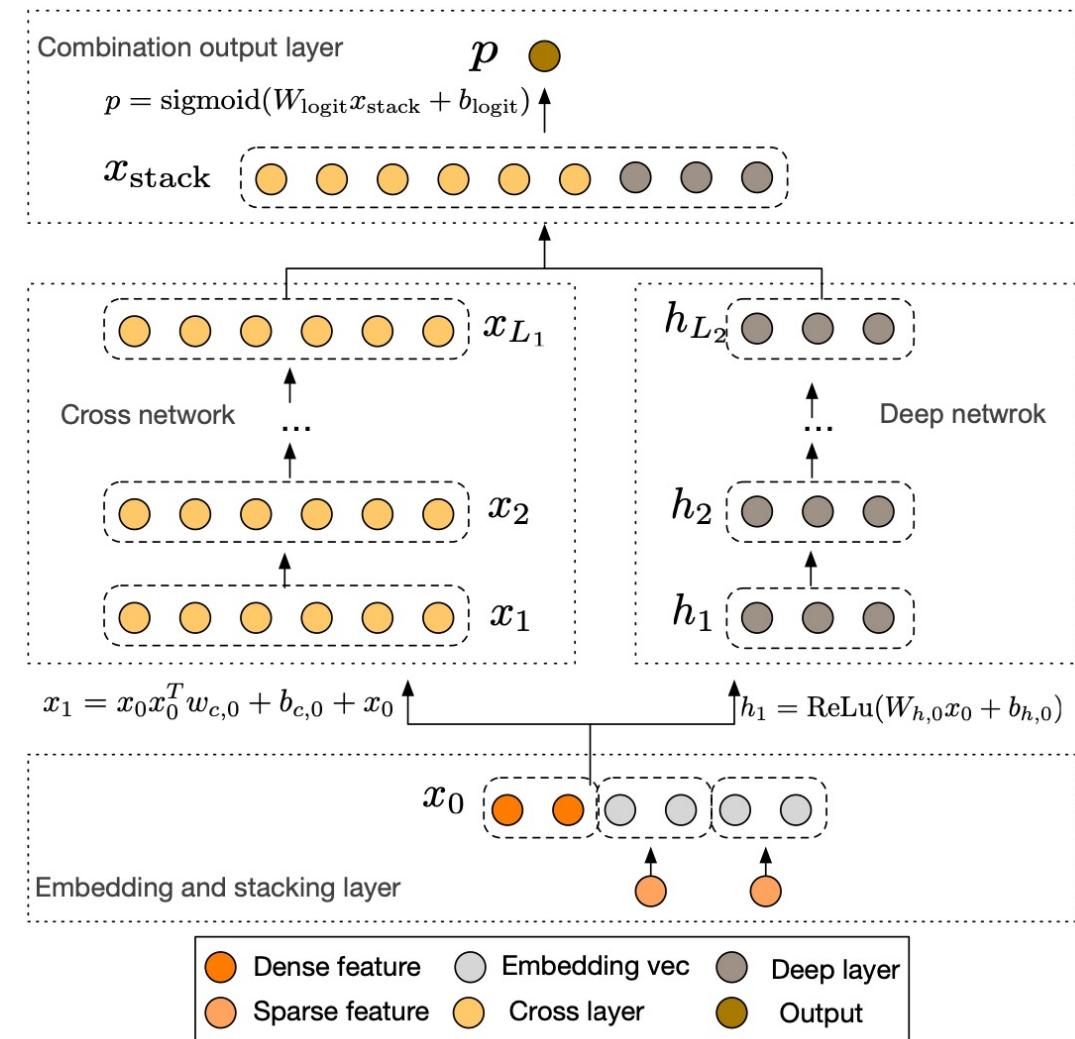
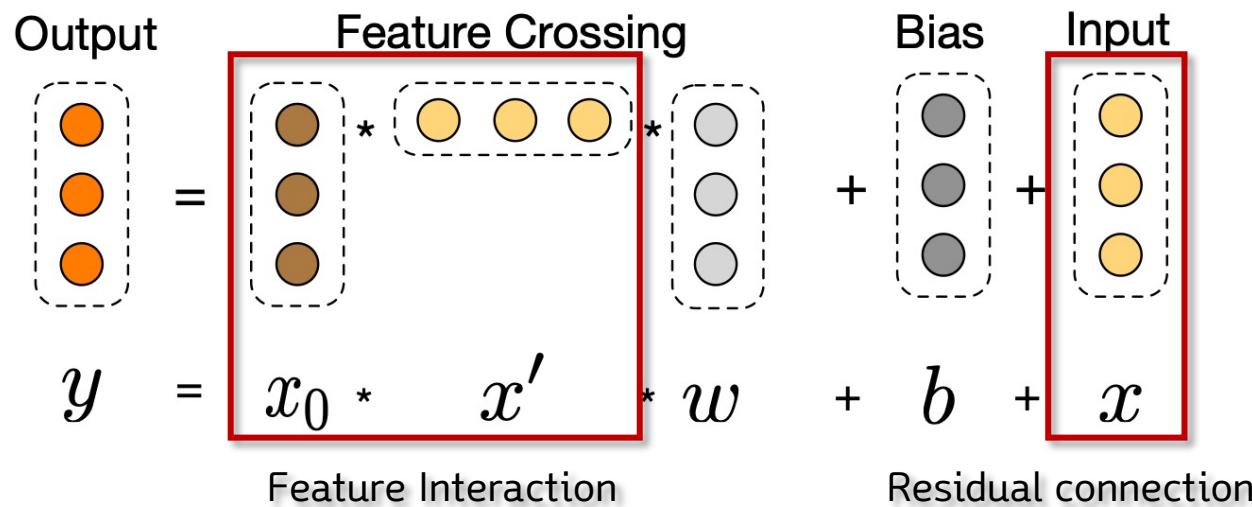
✓ 외적을 통해 3차원 텐서로 변환

✓ CNN 특성을 반영  
✓ 2차원 텐서를 벡터로 변환

✓ RNN 특성을 반영  
✓ 벡터 단위로 원소를 합침

# Deep & Cross Network (DCN)

- ✓ 단순히 모델을 깊게하는 것은 에러율을 높임  
→ ResNet의 경우 이를 Residual connection으로 해결
- ✓ 높은 차원의 interaction을 효율적으로 해보자
- ✓ Cross Network 제안



## DCN vs DNN

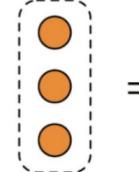
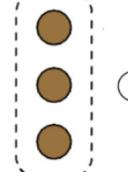
- ✓ 선형회귀, FM, DNN, DC(CrossLayer만 사용) 대비 가장 좋은 성능
- ✓ 아래 표의 결과로 CrossLayer가 효율적인 학습에 도움이 됨을 증명
- ✓ DNN을 사용하던 추천시스템의 경우 DCN을 사용하면 모델사이즈를  $\frac{1}{2}$ 로 줄일 수 있음

<b>Logloss</b>	0.4430	0.4460	0.4470	<b>0.4480</b>
DNN	$3.2 \times 10^6$	$1.5 \times 10^5$	$1.5 \times 10^5$	$7.8 \times 10^4$
DCN	$7.9 \times 10^5$	$7.3 \times 10^4$	$3.7 \times 10^4$	$3.7 \times 10^4$

파라미터를 절반만  
쓰고도 동일한 성능

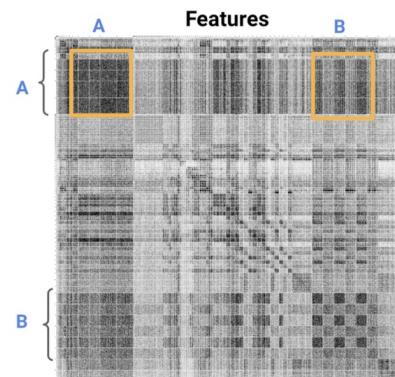
# DCNv2

- ✓ 모델의 Capacity를 늘리고자 CrossLayer의 파라미터를 2차원으로 확장

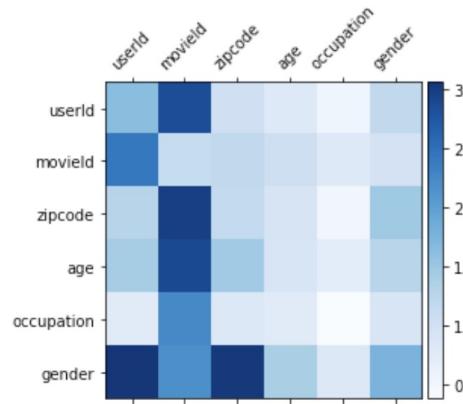
Output	Feature Crossing	Bias	Input
	$=$  $\odot$ $\left( \begin{array}{c c} \begin{matrix} \text{---} & \text{---} & \text{---} \\ \text{---} & \text{---} & \text{---} \\ \text{---} & \text{---} & \text{---} \end{matrix} & \begin{matrix} \text{---} \\ \text{---} \\ \text{---} \end{matrix} \\ \hline \begin{matrix} \text{---} & \text{---} & \text{---} \\ \text{---} & \text{---} & \text{---} \\ \text{---} & \text{---} & \text{---} \end{matrix} & \begin{matrix} \text{---} \\ \text{---} \\ \text{---} \end{matrix} \end{array} \times \begin{matrix} \text{---} \\ \text{---} \\ \text{---} \end{matrix} + \begin{matrix} \text{---} \\ \text{---} \\ \text{---} \end{matrix} \right) + \begin{matrix} \text{---} \\ \text{---} \\ \text{---} \end{matrix}$	$\begin{matrix} \text{---} \\ \text{---} \\ \text{---} \end{matrix}$	$\begin{matrix} \text{---} \\ \text{---} \\ \text{---} \end{matrix}$

$$x_{i+1} = x_0 \odot (W \times x_i + b) + x_i$$

- ✓ CrossLayer를 시각화하면 feature간 interaction의 정도를 파악 가능



(a) Production data

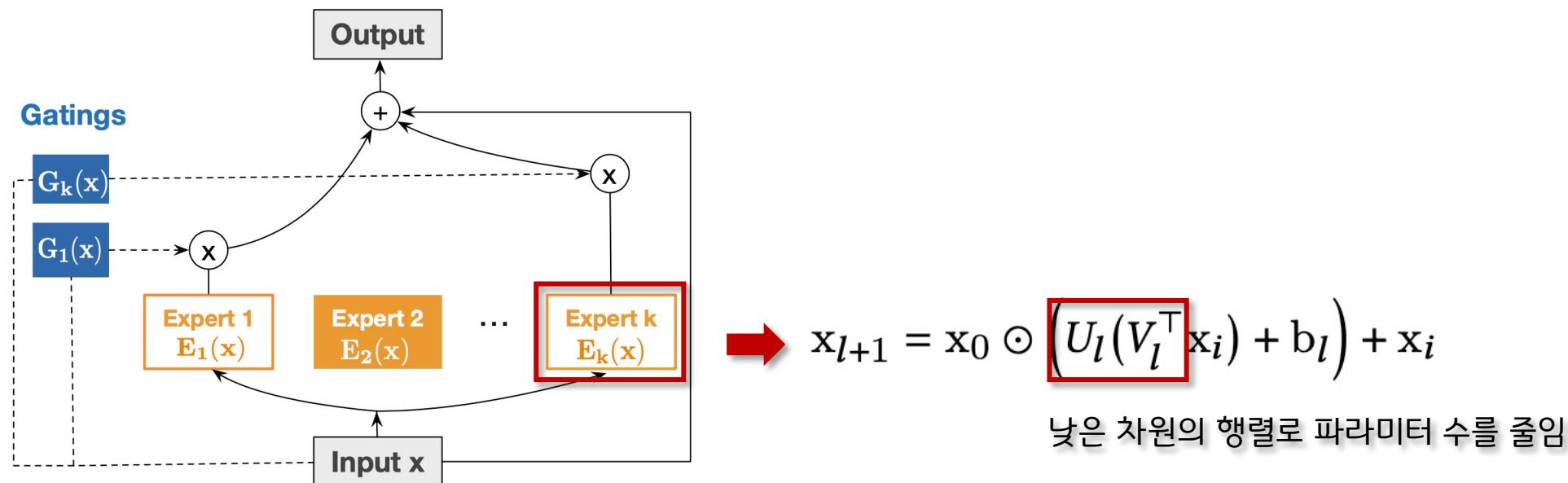


(b) MovieLens-1M

# DCN-Mix

## ✓ MoE + Low-rank Expert

- **MoE?** Multi-task Learning 구조
- **Low-rank Expert?** CrossNet의 가중치 행렬을 SVD로 분해하고 특이값을 1로 근사하여 압축



### 3 Research Trend

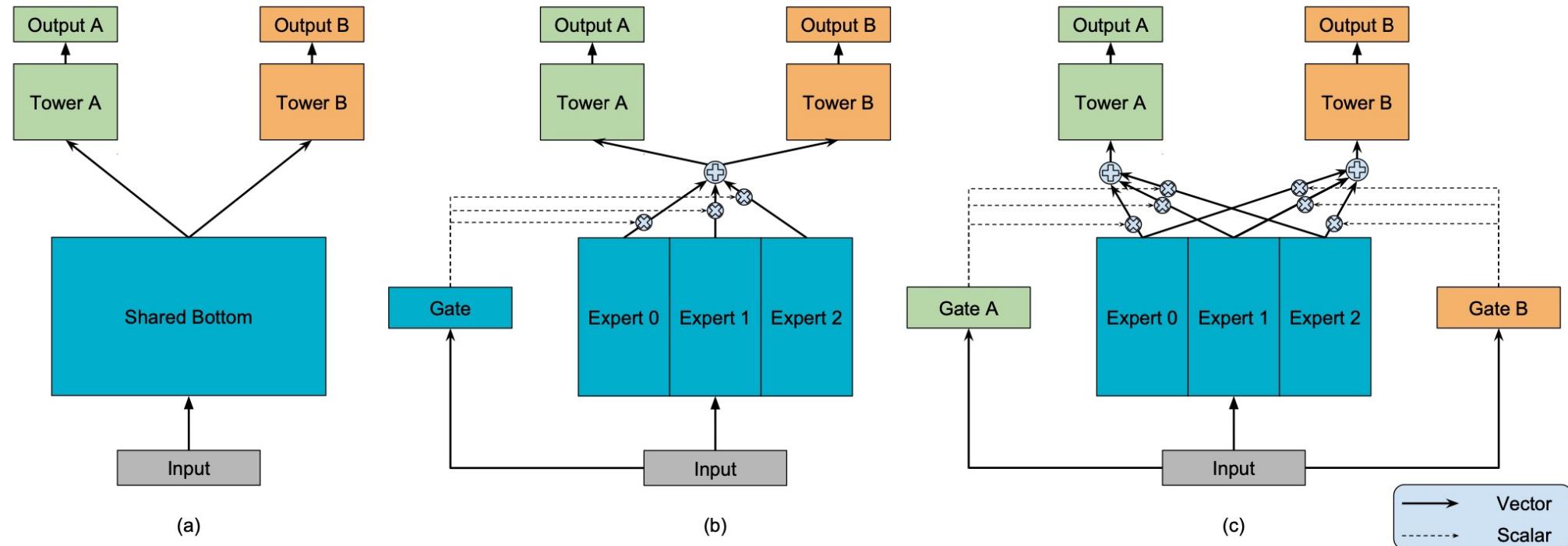
## DCNv2: 성능평가

- ✓ CrossLayer에 레이어당 파라미터수가 많아졌고 레이어 수를 줄였을 때 동일 파라미터수 대비 더 높은 성능을 보임
- ✓ DCN-Mix의 경우 DCN과 동일한 파라미터 수/연산으로 더 좋은 성능을 냄
- ✓ Model capacity가 높음이 증명되어 데이터 스케일이 클 경우 유리한 모델

Baseline	Criteo						MovieLens-1M			
	Logloss	AUC	Params	FLOPS	Best Setting		Logloss	AUC	Params	FLOPS
PNN	0.4421 (5.8E-4)	0.8099 (6.1E-4)	3.1M	6.1M	(3, 1024)	OPNN	0.3182 (1.4E-3)	0.8955 (3.3E-4)	54K	110K
DeepFm	0.4420 (1.4E-4)	0.8099 (1.5E-4)	1.4M	2.8M	(2, 768)	–	0.3202 (1.0E-3)	0.8932 (7.7E-4)	46K	93K
DLRM	0.4427 (3.1E-4)	0.8092 (3.1E-4)	1.1M	2.2M	(2, 768)	[512,256,64]	0.3245 (1.1E-3)	0.8890 (1.1E-3)	7.7K	16K
xDeepFm	0.4421 (1.6E-4)	0.8099 (1.8E-4)	3.7M	32M	(3, 1024)	$l=2, n=100$	0.3251 (4.3E-3)	0.8923 (8.6E-4)	160K	990K
AutoInt+	0.4420 (5.7E-5)	0.8101 (2.6E-5)	4.2M	8.7M	(4, 1024)	$l=2, h=2, e=40$	0.3204 (4.4E-4)	0.8928 (3.9E-4)	260K	500K
DCN	0.4420 (1.6E-4)	0.8099 (1.7E-4)	2.1M	4.2M	(2, 1024)	$l=4$	0.3197 (1.9E-4)	0.8935 (2.1E-4)	110K	220K
DNN	0.4421 (6.5E-5)	0.8098 (5.9E-5)	3.2M	6.3M	(3, 1024)	–	0.3201 (4.1E-4)	0.8929 (2.3E-4)	46K	92K
<b>Ours</b>										
DCN-V2	<b>0.4406 (6.2E-5)</b>	<b>0.8115 (7.1E-5)</b>	3.5M	7.0M	(2, 768)	$l=2$	0.3170 (3.6E-4)	0.8950 (2.7E-4)	110K	220K
DCN-Mix	0.4408 (1.0E-4)	0.8112 (9.8E-5)	2.4M	4.8M	(2, 512)	$l=3, K=4, r=258$	<b>0.3160 (4.9E-4)</b>	<b>0.8964 (2.9E-4)</b>	110K	210K
CrossNet	0.4413 (2.5E-4)	0.8107 (2.4E-4)	2.1M	4.2M	–	$l=4, K=4, r=258$	0.3185 (3.0E-4)	0.8937 (2.7E-4)	65K	130K

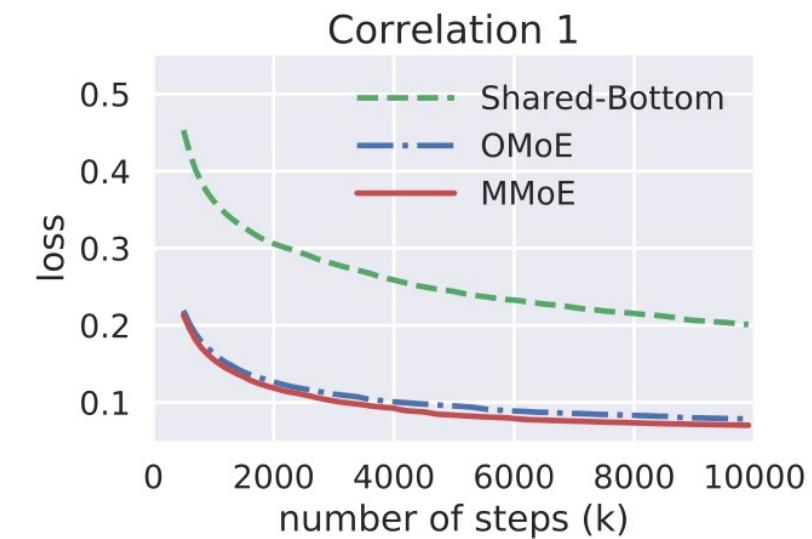
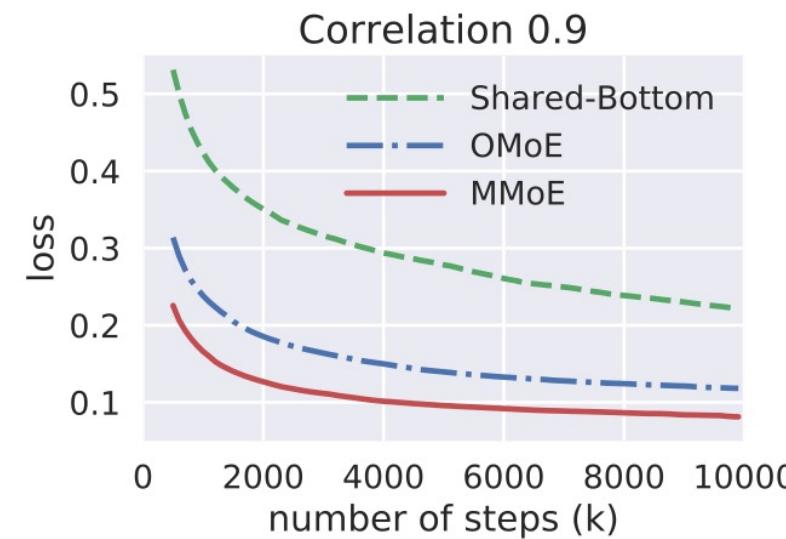
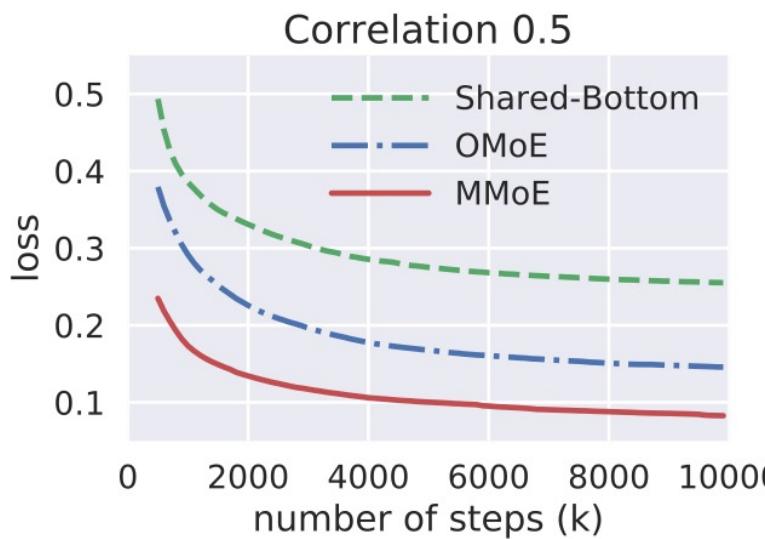
# Multi-task Learning with Multi-gate Mixture-of-Experts

- ✓ 하나의 태스크를 수행하기 위해서 연관되는 작은 태스크들을 반영해야함
  - 영상 추천의 경우, 썸네일이 자극적이면 클릭율이 높지만 좋아요, 시청 시간은 낮을 수 있음
- ✓ 데이터간의 상관관계가 낮으면 성능이 떨어질 수 있음 → 모델을 따로 학습하여 합치자



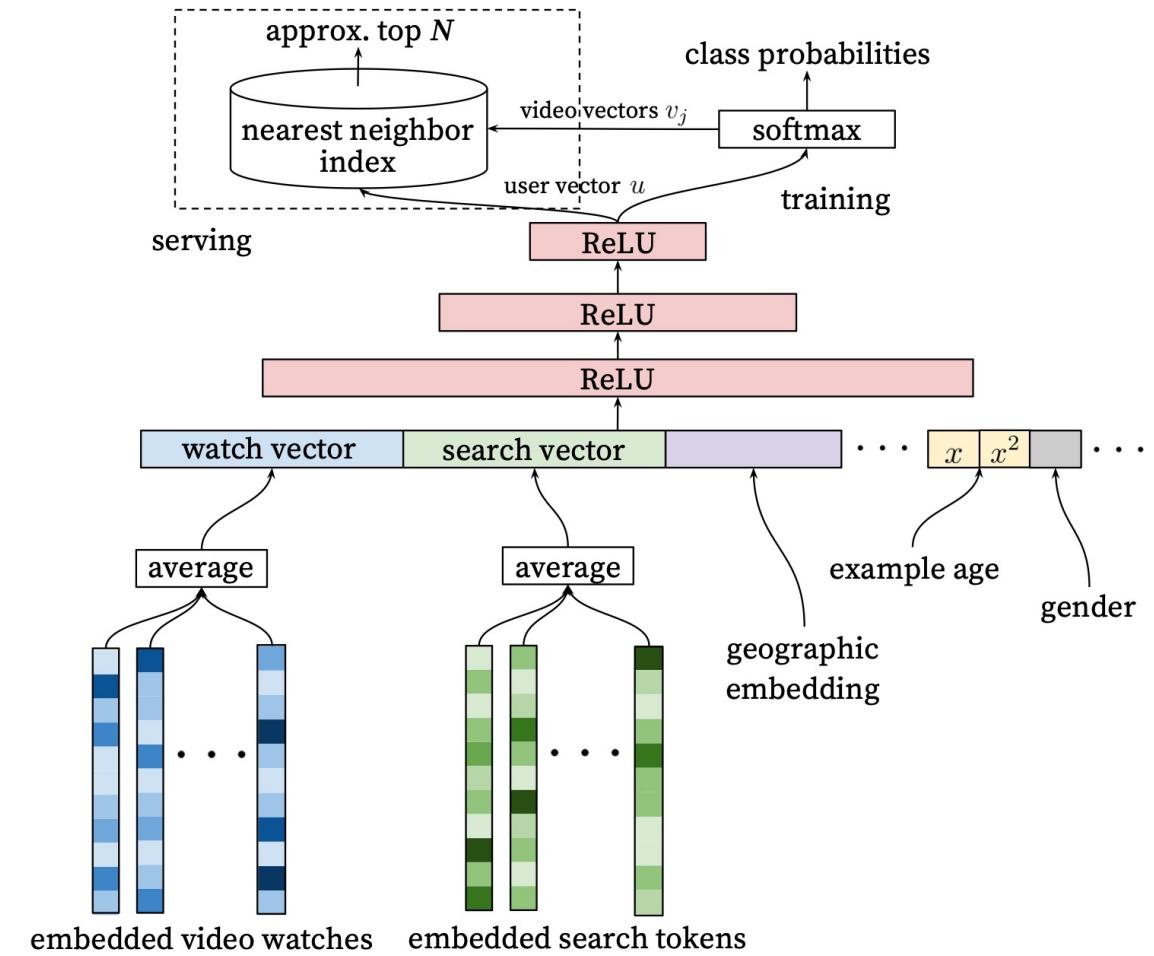
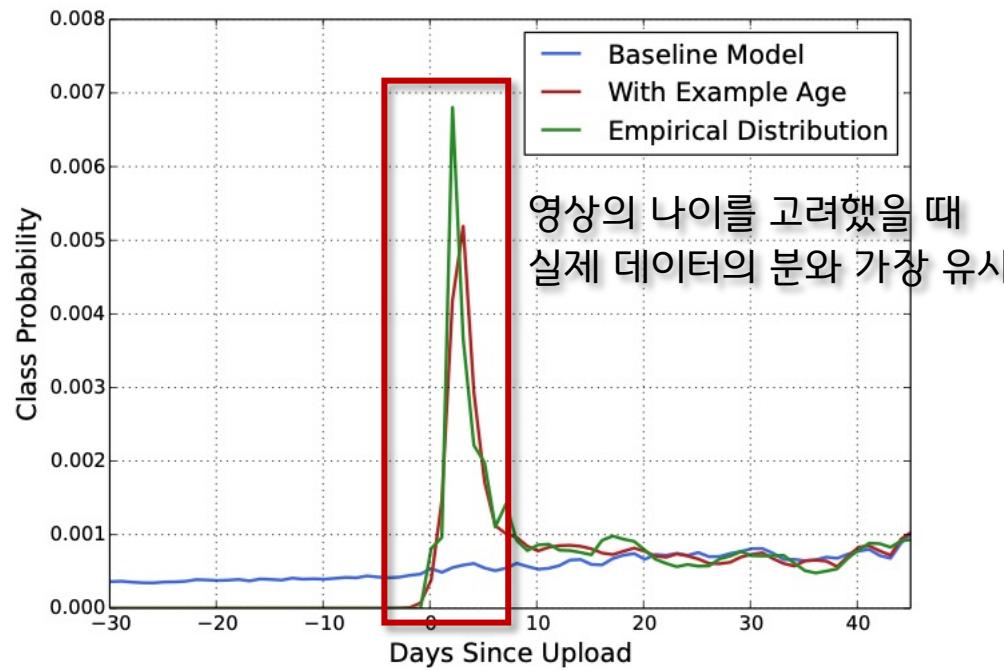
## MMoE: 성능평가

- ✓ 상관관계가 높은 경우에도 성능이 좋음
- ✓ 상관관계가 낮은 경우에는 Gate를 Task-dependant하게 두는 것이 성능이 좋음
- ✓ 다양한 지표들을 활용하여 예측하는 경우 MMoE를 사용하면 좋을 것



# Deep Neural Networks for Youtube Recommendations

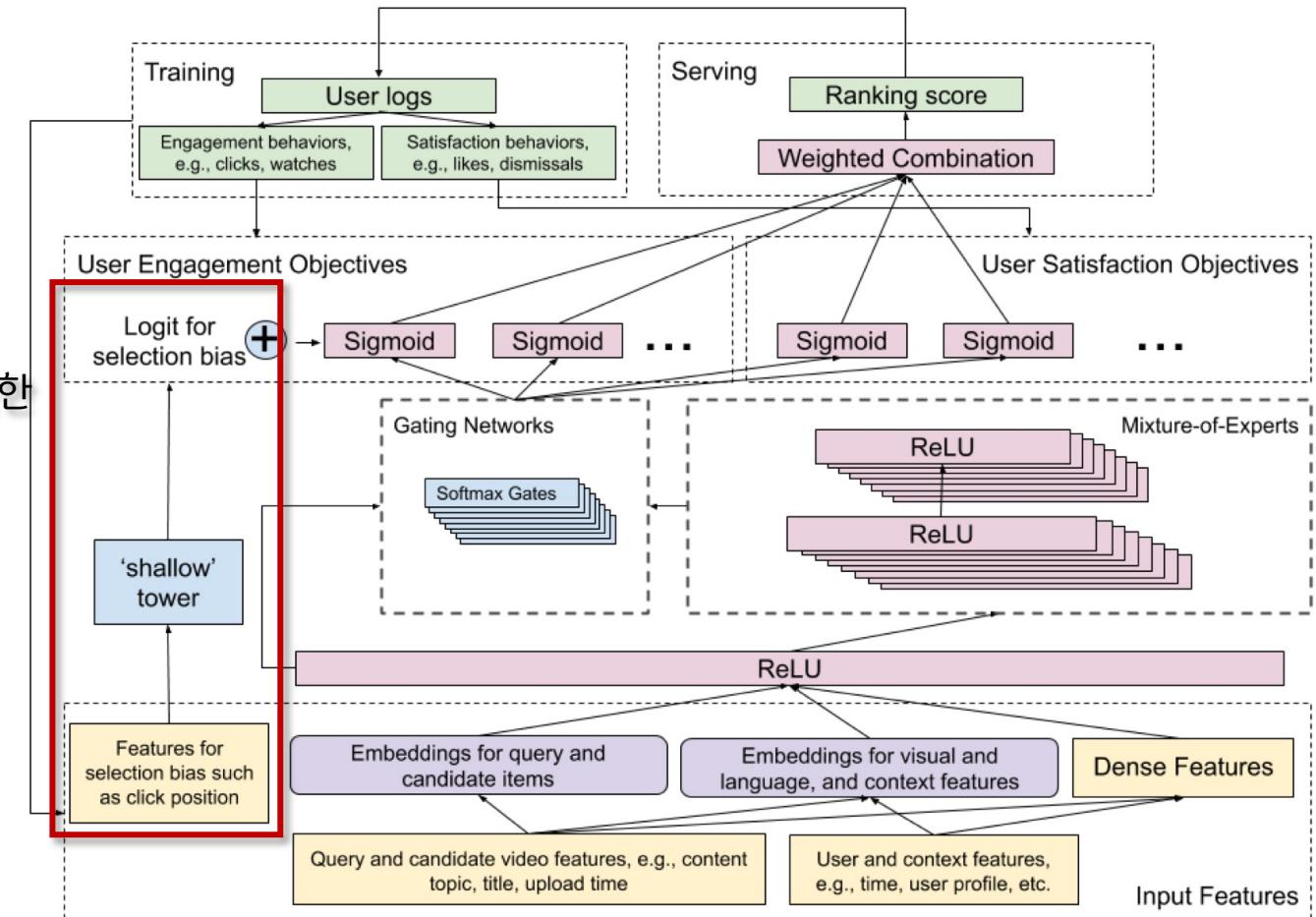
- ✓ 2-Stage 모델의 Candidate Generation
- ✓ 유저 임베딩 + 영상 임베딩 → 유사도 기반 탐색
- ✓ Feature Engineering의 중요성



## Recommending What Video to Watch Next - A Multitask Ranking System

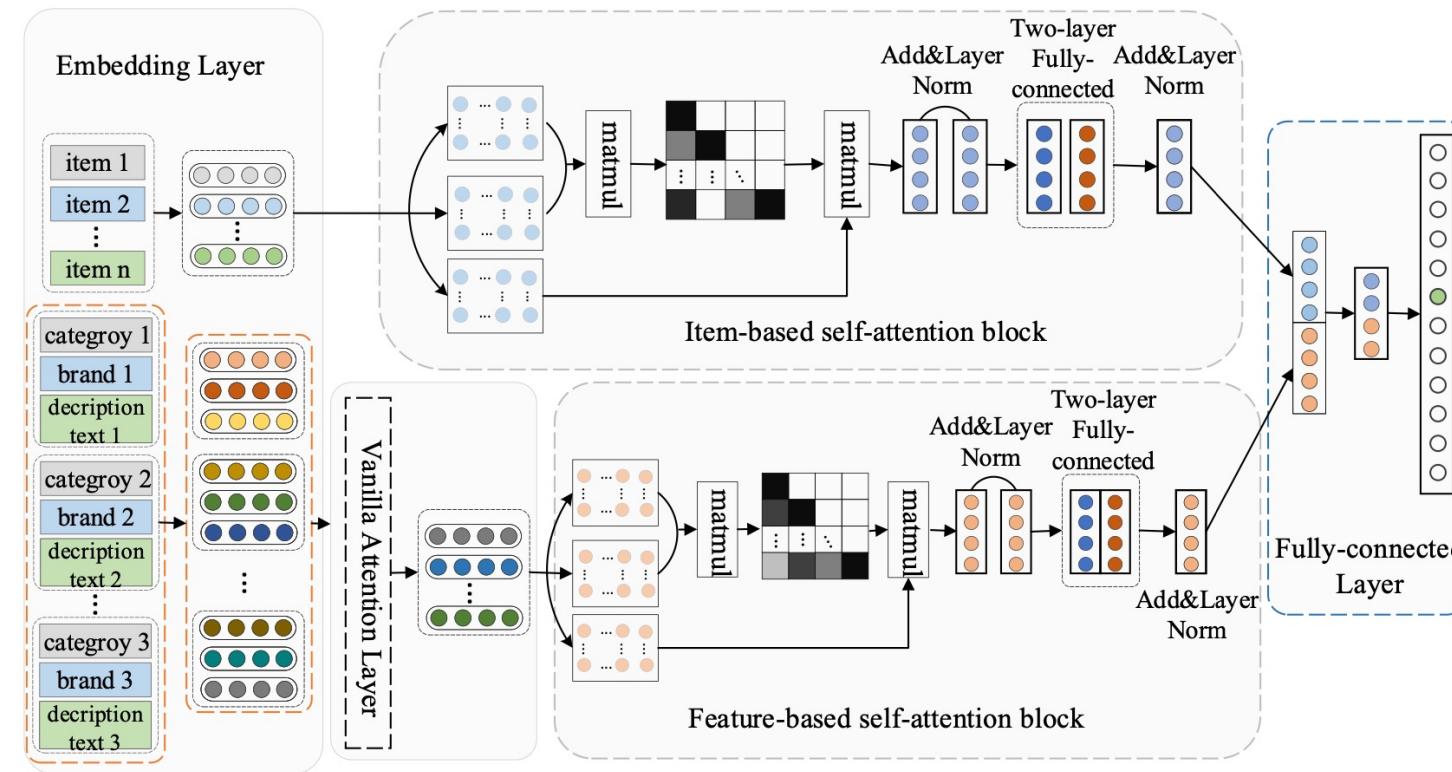
- ✓ 2-Stage 모델의 Ranking
- ✓ 다양한 태스크로 최적의 랭킹을 하고자 함
  - Engagement objectives: 클릭, 시청 시간
  - Satisfaction objectives: 좋아요, 순위 평가
- ✓ 편향되어 학습되는 것을 방지하고자 함

상위 노출로 인한  
편향성 제거



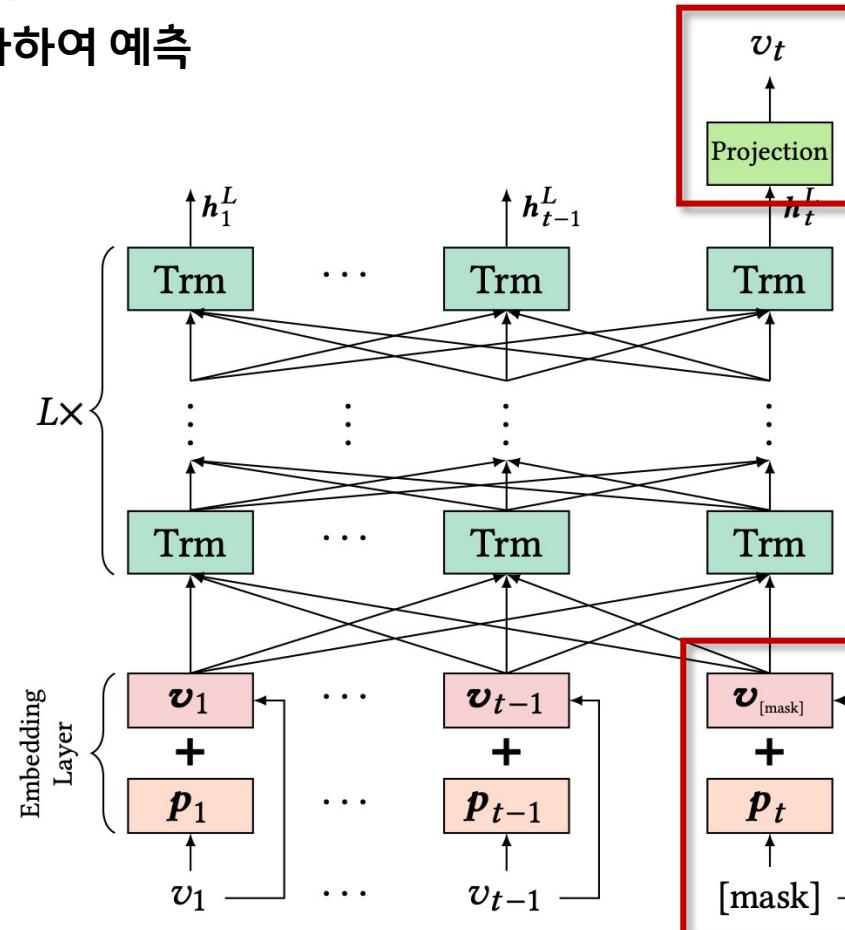
# Feature-level Deeper Self-Attention

- ✓ RNN기반의 시퀀스 예측은 Long-term dependency, 병렬화 X, 시퀀스 중요도 반영 X  
→ Transformer 구조를 사용하여 해결
- ✓ 상품 구매에 대한 시퀀스 + 다양한 상품 정보(카테고리, 브랜드, 텍스트)를 임베딩



# BERT4Rec

- ✓ BERT의 Language Token을 item으로 대체
- ✓ 아이템과 시간에 대한 임베딩을 입력
- ✓ 마지막 시퀀스에 마스킹 토큰을 추가하여 예측



### 3 Research Trend

# Sequential Recommendation with Transformer

- ✓ BERT의 Pre-training 방식이 효과적
- ✓ 상품정보를 추가로 반영하는 것이 효과적
- ✓ 두가지 방법을 모두 사용하면 더 큰 성능향상 예상
- ✓ 세션이 다양하고 시퀀스가 긴 경우에 적절할 것

Dataset	Method	@5		@10	
		Hit	NDCG	Hit	NDCG
Tmall	PopRec	0.1532	0.0988	0.2397	0.1267
	BPR	0.1749	0.1129	0.2647	0.1418
	FPMC	0.2731	0.2034	0.3680	0.2339
	TransRec	0.2652	0.1854	0.3773	0.2214
	GRU4Rec	0.1674	0.1217	0.2446	0.1465
	CSAN	0.3481	0.2440	0.4787	0.2863
	<b>SASRec</b>	<b>0.3572</b>	<b>0.2531</b>	<b>0.4840</b>	<b>0.2940</b>
	SASRec+	0.3427	0.2415	0.4714	0.2829
	SASRec++	0.3550	0.2534	0.4785	0.2932
	CFSA	0.3836	0.2724	0.5152	0.3149
	<b>FDSA</b>	<b>0.3940</b>	<b>0.2820</b>	<b>0.5197</b>	<b>0.3226</b>

Datasets	Metric	POP	BPR-MF	NCF	FPMC	GRU4Rec	GRU4Rec <sup>+</sup>	Caser	SASRec	BERT4Rec	Improv.
ML-20m	HR@1	0.0221	0.0553	0.0231	0.1079	0.1459	0.2021	0.1232	<u>0.2544</u>	<b>0.3440</b>	35.22%
	HR@5	0.0805	0.2128	0.1358	0.3601	0.4657	0.5118	0.3804	<u>0.5727</u>	<b>0.6323</b>	10.41%
	HR@10	0.1378	0.3538	0.2922	0.5201	0.5844	0.6524	0.5427	<u>0.7136</u>	<b>0.7473</b>	4.72%
	NDCG@5	0.0511	0.1332	0.0771	0.2239	0.3090	0.3630	0.2538	<u>0.4208</u>	<b>0.4967</b>	18.04%
	NDCG@10	0.0695	0.1786	0.1271	0.2895	0.3637	0.4087	0.3062	<u>0.4665</u>	<b>0.5340</b>	14.47%
	MRR	0.0709	0.1503	0.1072	0.2273	0.2967	0.3476	0.2529	<u>0.4026</u>	<b>0.4785</b>	18.85%

Q / A