Wide & Deep Learning for Recommender Systems

By Heng-Tze Cheng et al.

발표자 김이삭

Contents

Introduction

- Memorization

-feature 또는 item의 빈번한 공존을 학습하고 historical data에서 이용가능한 상관관계를 이용

-Wide set of cross-product feature transformation 사용

- Generalization

-correlation의 transitivity에 근거하고 과거에 발생한적이 없는 새로운 feature combination을 탐색하는 것

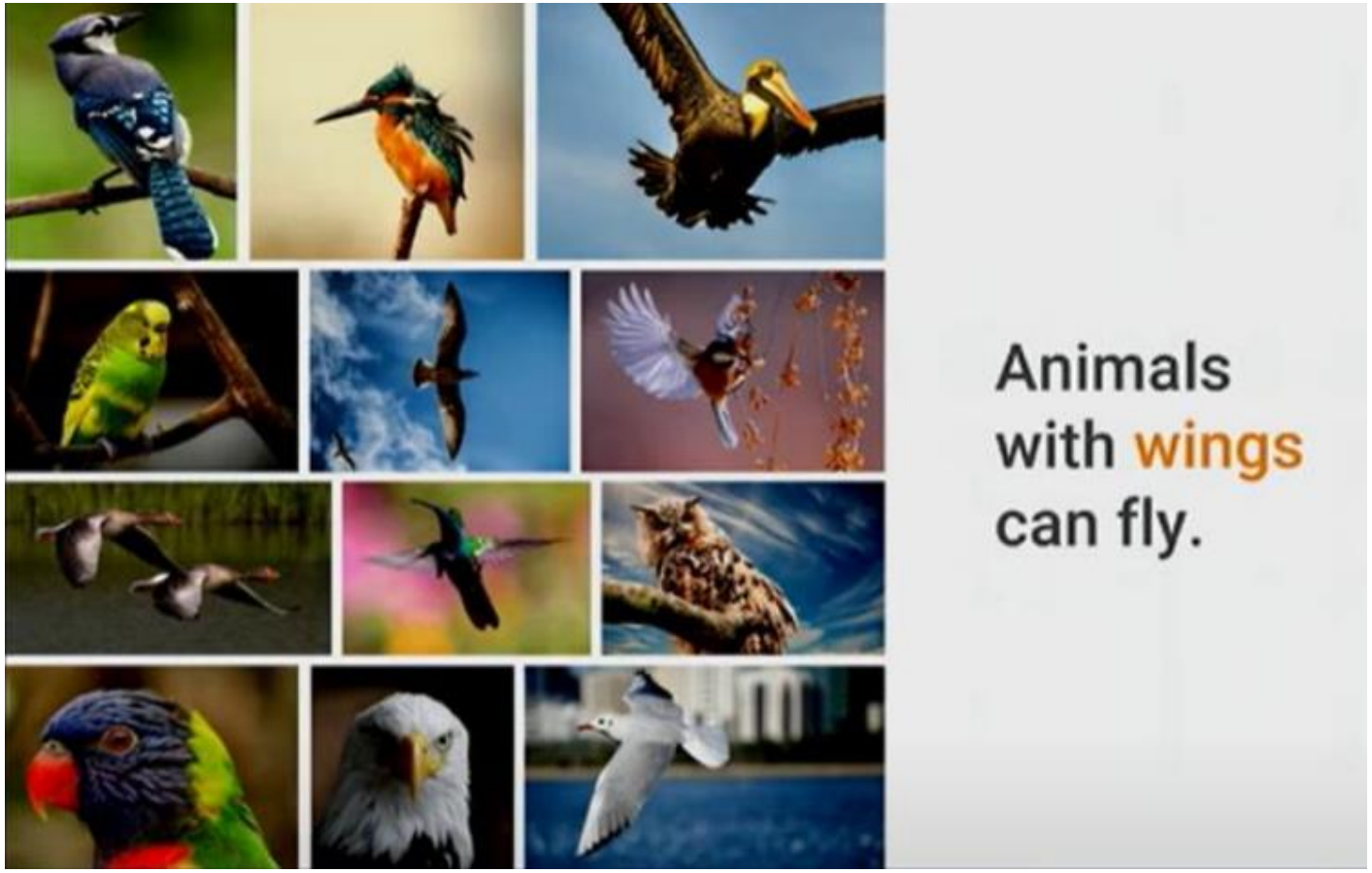
-Deep Neural Network 사용

Introduction

- Memorization

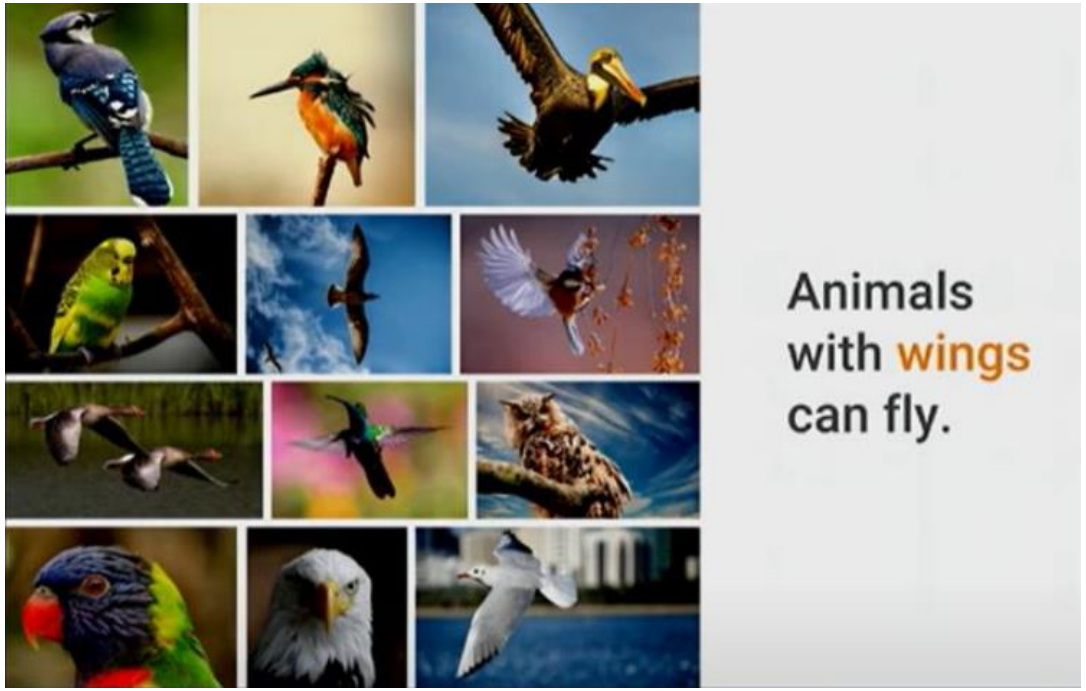
Introduction

- Generalization



Introduction

- Generalization + Memorization



exception

Introduction
Memorization

- 주로 logistic regression(linear model) 사용 – 확장 가능성, 단순함
- One-hot encoding → binary sparse feature로 학습
- Cross product사용 → feature pair의 공존 표현

- 수동 feature engineering 필요
- 이미 나타난 pair에 대해서만 학습 가능

Introduction
Generalization

- FM이나 DNN 사용
- Feature data 를 low dim dense embedding vector 로 학습

→이전에 나타나지 않은 data 예측 가능

- Feature engineering 부하 감소

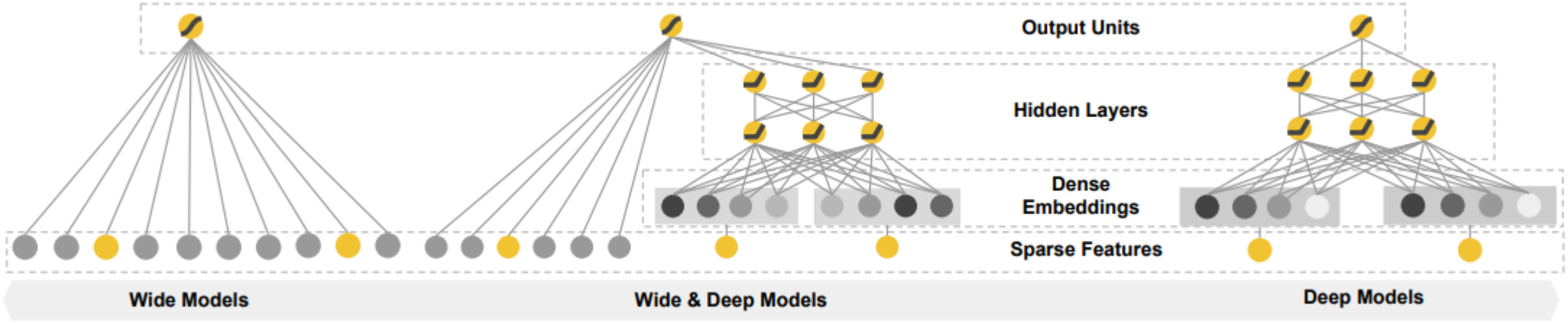
- Sparse, high rank data인 경우 학습 어려움

Introduction

• Wide & Deep Learning

Wide linear model + Deep neural network

→Memorization + Generalization



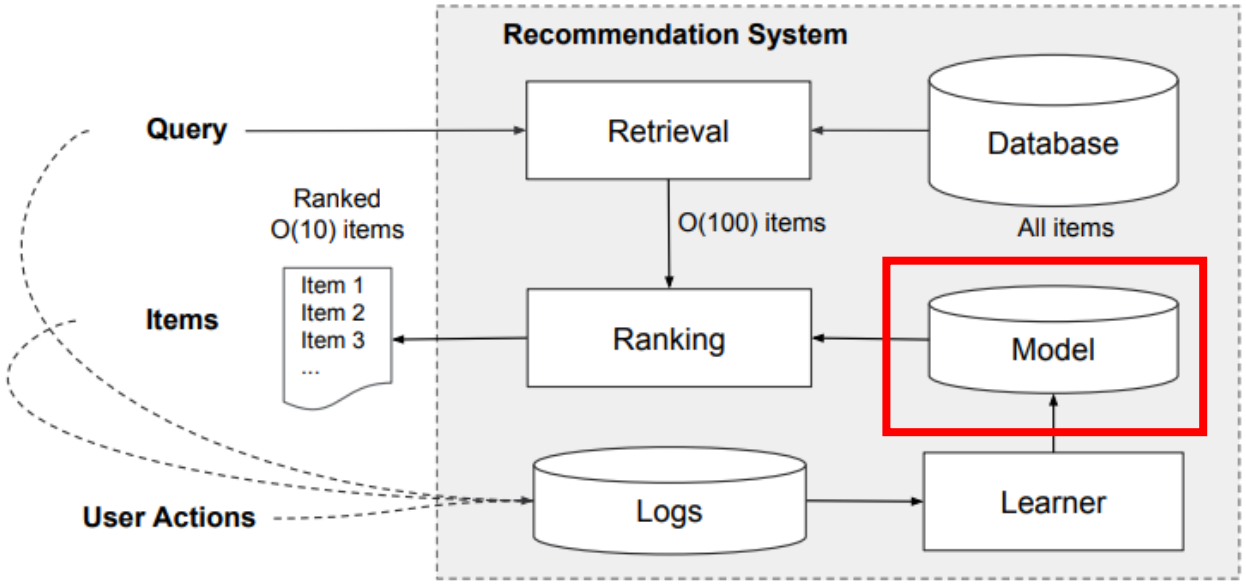
Recommender System Overview

- P(y|x)를 기반으로 앱 순위 결정

y: action

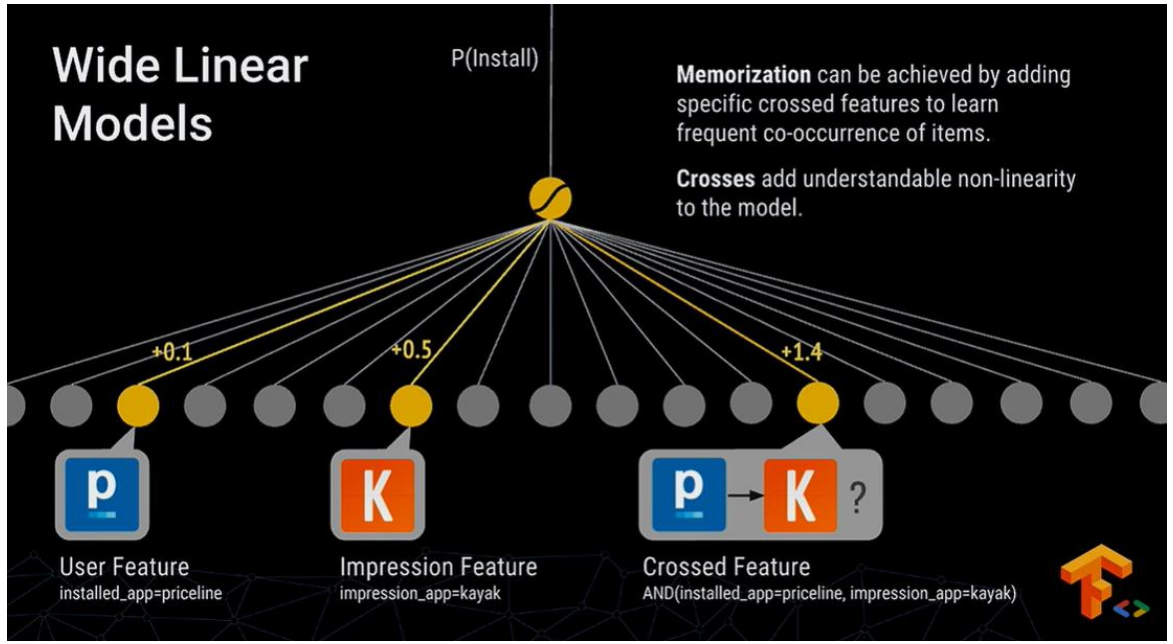
x: user features, contextual, impression features

- Wide & Deep 모델 활용



Wide & Deep Learning

wide part



$y = w^T x + b$

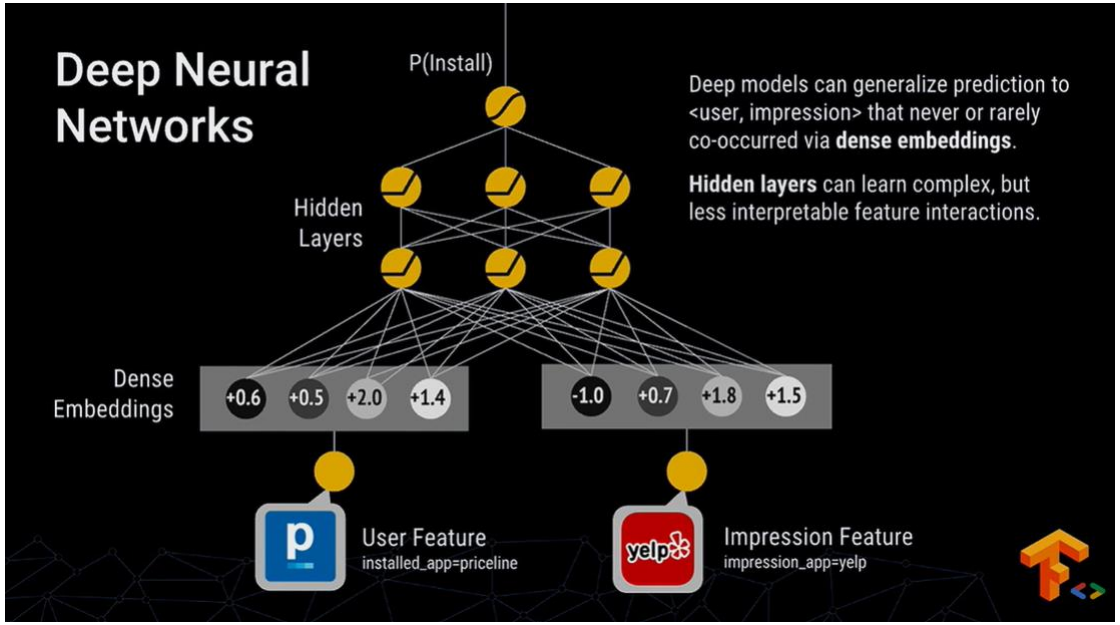
$$\phi_k(\mathbf{x}) = \prod_{i=1}^{d} x_i^{c_{ki}} \quad c_{ki} \in \{0, 1\}$$

Cross product transformation

- Logistic regression
- 입력:User installed app, Impression app, cross product

Wide & Deep Learning

deep part



$$a^{(l+1)} = f(W^{(l)}a^{(l)} + b^{(l)})$$

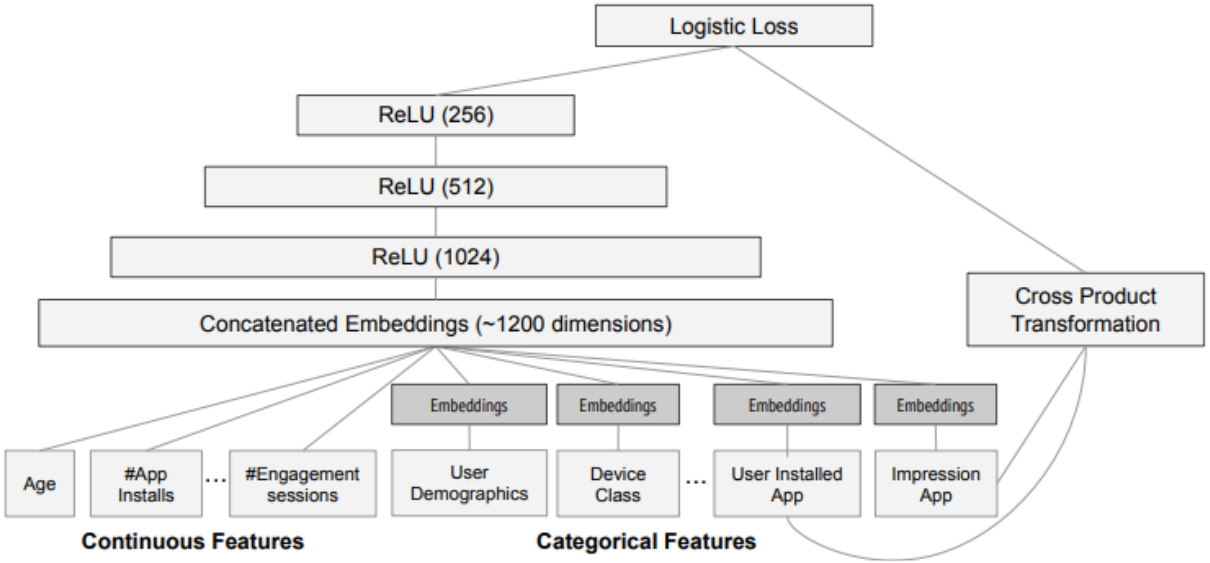
- Deep Neural Net
- 입력: Continuous and Categorical features

Age, #App Installs, Device Class, Installed app, Impression App 등등

Wide & Deep Learning
joint training

- Ensemble이 아닌 두 모델
한번에 back propagation

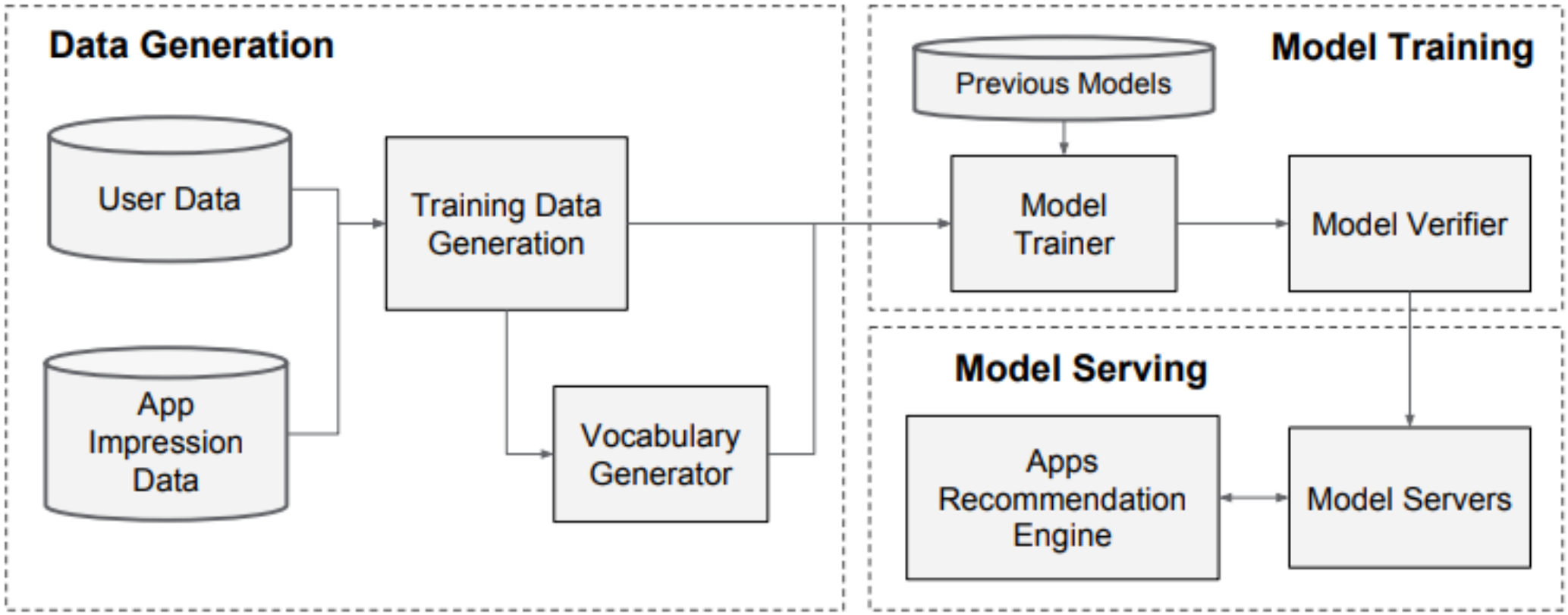
- Ensemble과 달리 두 모델의
파라미터를 동시에 최적화 가능



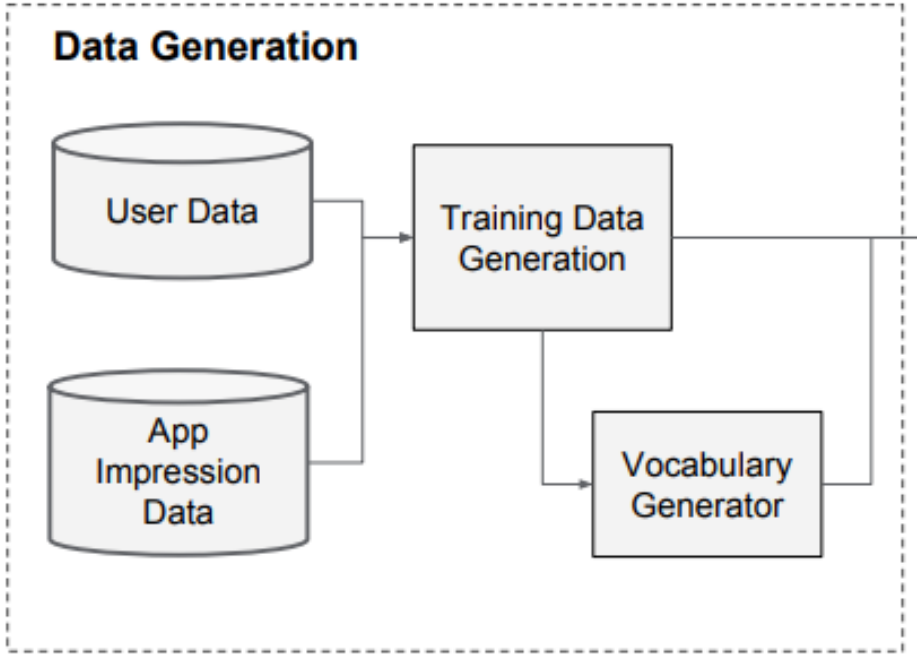
$$P(Y=1|\mathbf{x}) = \sigma(\mathbf{w}_{wide}^T[\mathbf{x}, \phi(\mathbf{x})] + \mathbf{w}_{deep}^T a^{(l_f)} + b)$$

- 더 적은 모델 사이즈로 구현 가능, 소수의 cross product로 deep 모델의 약점만 보완

System Implementation

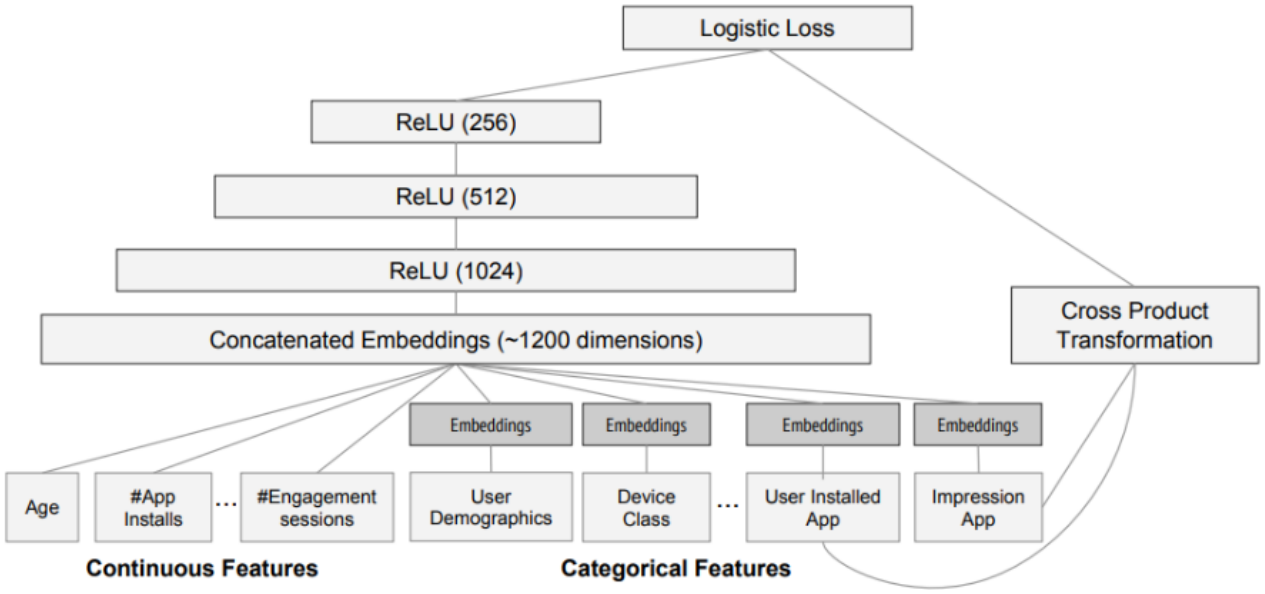


Data Generation



- 일정 기간 동안의 user data와 impression data를 사용해 training data 생성
- Vocabulary를 통해 categorical feature를 integer ID로 매핑

Model training



전체 모델 구조

매번 re-train을 하면 시간이 오래 걸리기 때문에 warm-starting system으로 학습

warm-starting system - 임베딩과 이전 모형의 선형 모형 가중치를 사용하여 새 모형 초기값을 설정

Model Serving

- 학습된 모델을 모델 서버에 업로드 후 request와 유저 정보에 맞춰 앱 랭킹 출력
- 소규모 batch를 병렬로 실행하여 성능 최적화

Experiments Results

Model	Offline AUC	Online Acquisition Gain
Wide (control)	0.726	0%
Deep	0.722	+2.9%
Wide & Deep	0.728	+3.9%

1% 사용자를 대상으로 테스트한 결과 유의미한 증가가 있었다.

Batch size	Number of Threads	Serving Latency (ms)
200	1	31
100	2	17
50	4	14

배치사이즈에 따른 Serving Latency

Implementation

- Dataset: Kaggle Adult income dataset

  15 features(age, workclass, race …)

  target: find income <=50k or not

  loss function – binary cross entropy

  optimizer – adam

  crossed features for wide model:
['education', 'occupation'], ['native_country', 'occupation']

Implementation

```
Epoch 1/10
509/509 [==============================] - 1s 2ms/step - loss: 0.4881 - accuracy: 0.7752
Epoch 2/10
509/509 [==============================] - 1s 2ms/step - loss: 0.3924 - accuracy: 0.8218
Epoch 3/10
509/509 [==============================] - 1s 2ms/step - loss: 0.3738 - accuracy: 0.8267
Epoch 4/10
509/509 [==============================] - 1s 2ms/step - loss: 0.3664 - accuracy: 0.8298
Epoch 5/10
509/509 [==============================] - 1s 2ms/step - loss: 0.3623 - accuracy: 0.8314
Epoch 6/10
509/509 [==============================] - 1s 2ms/step - loss: 0.3598 - accuracy: 0.8330
Epoch 7/10
509/509 [==============================] - 1s 2ms/step - loss: 0.3580 - accuracy: 0.8324
Epoch 8/10
509/509 [==============================] - 1s 2ms/step - loss: 0.3569 - accuracy: 0.8325
Epoch 9/10
509/509 [==============================] - 1s 2ms/step - loss: 0.3559 - accuracy: 0.8338
Epoch 10/10
509/509 [==============================] - 1s 2ms/step - loss: 0.3552 - accuracy: 0.8341
509/509 [==============================] - 1s 1ms/step - loss: 0.6571 - accuracy: 0.6861
wide model accuracy: 0.6860758066177368
```

```
Epoch 1/10
509/509 [==============================] - 3s 3ms/step - loss: 1.4082 - accuracy: 0.7884
Epoch 2/10
509/509 [==============================] - 2s 4ms/step - loss: 0.4257 - accuracy: 0.8253
Epoch 3/10
509/509 [==============================] - 2s 4ms/step - loss: 0.4080 - accuracy: 0.8331
Epoch 4/10
509/509 [==============================] - 2s 3ms/step - loss: 0.3977 - accuracy: 0.8368
Epoch 5/10
509/509 [==============================] - 2s 3ms/step - loss: 0.3925 - accuracy: 0.8373
Epoch 6/10
509/509 [==============================] - 2s 3ms/step - loss: 0.3908 - accuracy: 0.8360
Epoch 7/10
509/509 [==============================] - 2s 3ms/step - loss: 0.3878 - accuracy: 0.8367
Epoch 8/10
509/509 [==============================] - 2s 3ms/step - loss: 0.3875 - accuracy: 0.8366
Epoch 9/10
509/509 [==============================] - 2s 3ms/step - loss: 0.3886 - accuracy: 0.8359
Epoch 10/10
509/509 [==============================] - 2s 3ms/step - loss: 0.3851 - accuracy: 0.8394
509/509 [==============================] - 1s 2ms/step - loss: 0.9093 - accuracy: 0.7374
deep model accuracy: 0.7374240159988403
```

```
Epoch 1/10
255/255 [==============================] - 2s 4ms/step - loss: 1.8503 - accuracy: 0.7709
Epoch 2/10
255/255 [==============================] - 1s 5ms/step - loss: 0.4104 - accuracy: 0.8265
Epoch 3/10
255/255 [==============================] - 1s 4ms/step - loss: 0.3775 - accuracy: 0.8407
Epoch 4/10
255/255 [==============================] - 1s 4ms/step - loss: 0.3626 - accuracy: 0.8440
Epoch 5/10
255/255 [==============================] - 1s 4ms/step - loss: 0.3555 - accuracy: 0.8463
Epoch 6/10
255/255 [==============================] - 1s 4ms/step - loss: 0.3491 - accuracy: 0.8483
Epoch 7/10
255/255 [==============================] - 1s 4ms/step - loss: 0.3455 - accuracy: 0.8494
Epoch 8/10
255/255 [==============================] - 1s 4ms/step - loss: 0.3421 - accuracy: 0.8518
Epoch 9/10
255/255 [==============================] - 1s 4ms/step - loss: 0.3412 - accuracy: 0.8510
Epoch 10/10
255/255 [==============================] - 1s 4ms/step - loss: 0.3377 - accuracy: 0.8513
509/509 [==============================] - 1s 2ms/step - loss: 0.7132 - accuracy: 0.7323
wide and deep model accuracy: 0.7322645783424377
```

Accuracy
Wide: 0.686
Deep: 0.737
Wide & deep: 0.732

Related Work

- Factorization Machine에서 두 변수간 상호작용을 일반화하는 방법
- RNN에서 복잡성을 줄이기 위해 제안된 joint training 활용
- CF 기반과 달리 앱 추천 시스템의 사용자 및 impression data에 대한 wide & deep model을 jointly train

Conclusion

- Wide linear model은 cross-product feature transformations을 사용하여 sparse feature interaction을 효과적으로 memorize할 수 있다
- DNN은 low-dim embedding을 통해 이전에 본적 없는 feature interactions으로 일반화할 수 있다.
- 두 모델의 장점을 결합한 wide & deep learning framework를 제안했고 큰 성능 개선이 있었다.