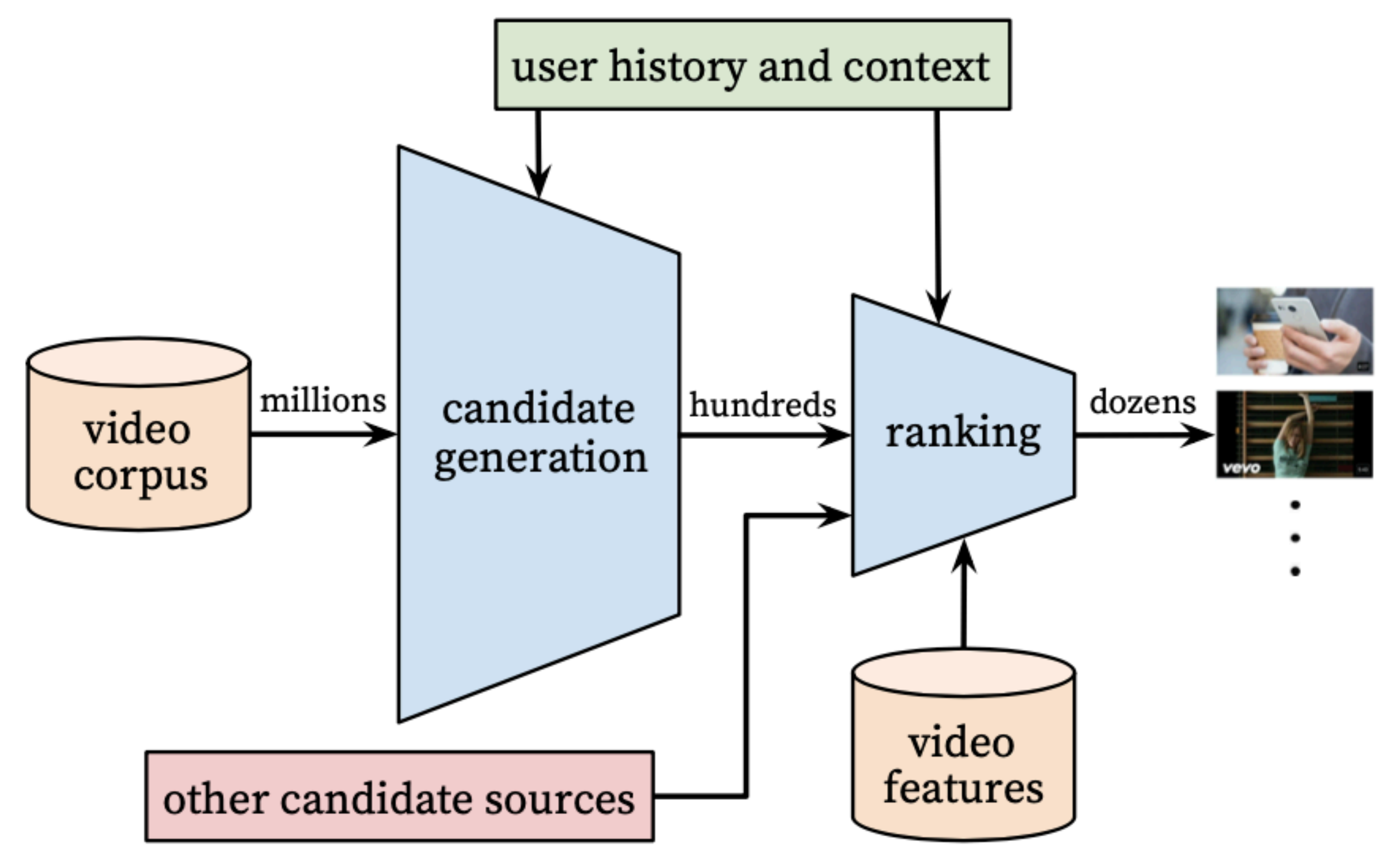
YouTube Recommendations

SoJeong Lee, 2020

overview



Scale Freshness Noise

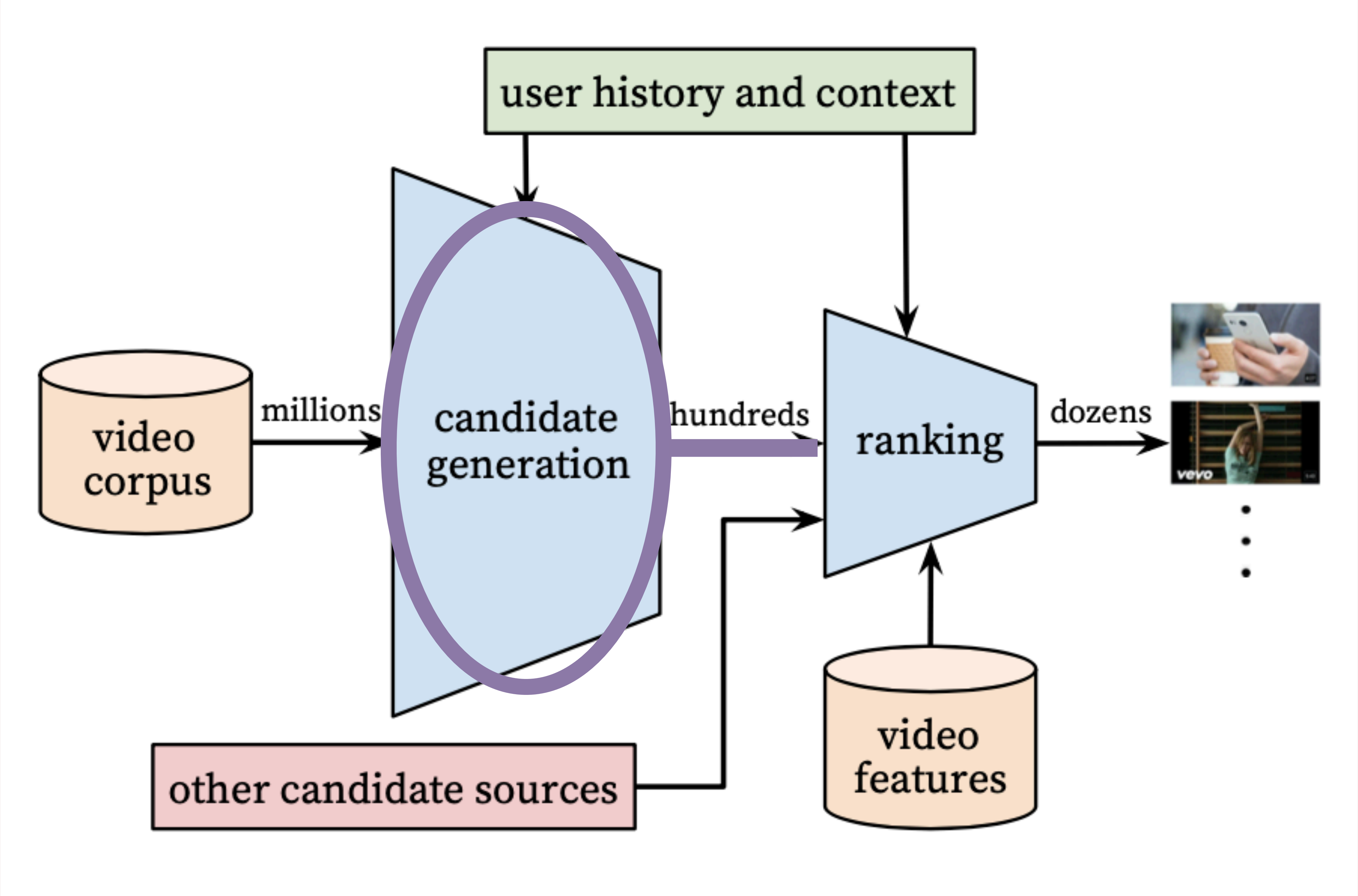
Candidate Generation

- Recommendation as classification
- Model Architecture
- Heterogeneous Signals
- Label and Context Selection
- Experiments with Features and Depth

Ranking

- Feature Representation
- Modeling Expected Watch Time
- Experiments with Hidden Layers

Candidate Generation



user와 관련된 백단위의 후보군 영상으로 줄이는 과정

user의 이전 watches를 embedding한 network
rank loss에 의해 학습되는 matrix fatorization → non-linear generalization한 matrix fatorization

Candidate Generation

Recommendation as Classification

“extreme multiclass classification”

$$P(w_t = i|U, C) = \frac{e^{v_i u}}{\sum_{j \in V} e^{v_j u}}$$

특정 시간*(t)*에 사용자*(U)*가 *C(context)*를 가지고 있을 때,

수백만 개의 비디오*(i)*를 볼 확률

---

training data

pair(user, context) / candidate videos $\xrightarrow[\text{embedding}]{}$ vector

implicit feedback[영상 끝까지 본 경우 -> positive]

Heterogeneous signals

example age …

Candidate Generation

Recommendation as Classification

To train

문제 the number of Classes ↑ in Softmax Classification ⟶ 계산량 증가

해결책! negative class sampling $\xrightarrow[\text{보정}]{}$ importance weighting

≒ ~~hierarchical softmax~~ 연관없는 class끼리도 분류하려 노력
비슷한 것 끼리 분류하기 어려워짐

cross-entropy loss : true label과 negative class에 의해 minimized

Candidate Generation

Recommendation as Classification

In Serving Time

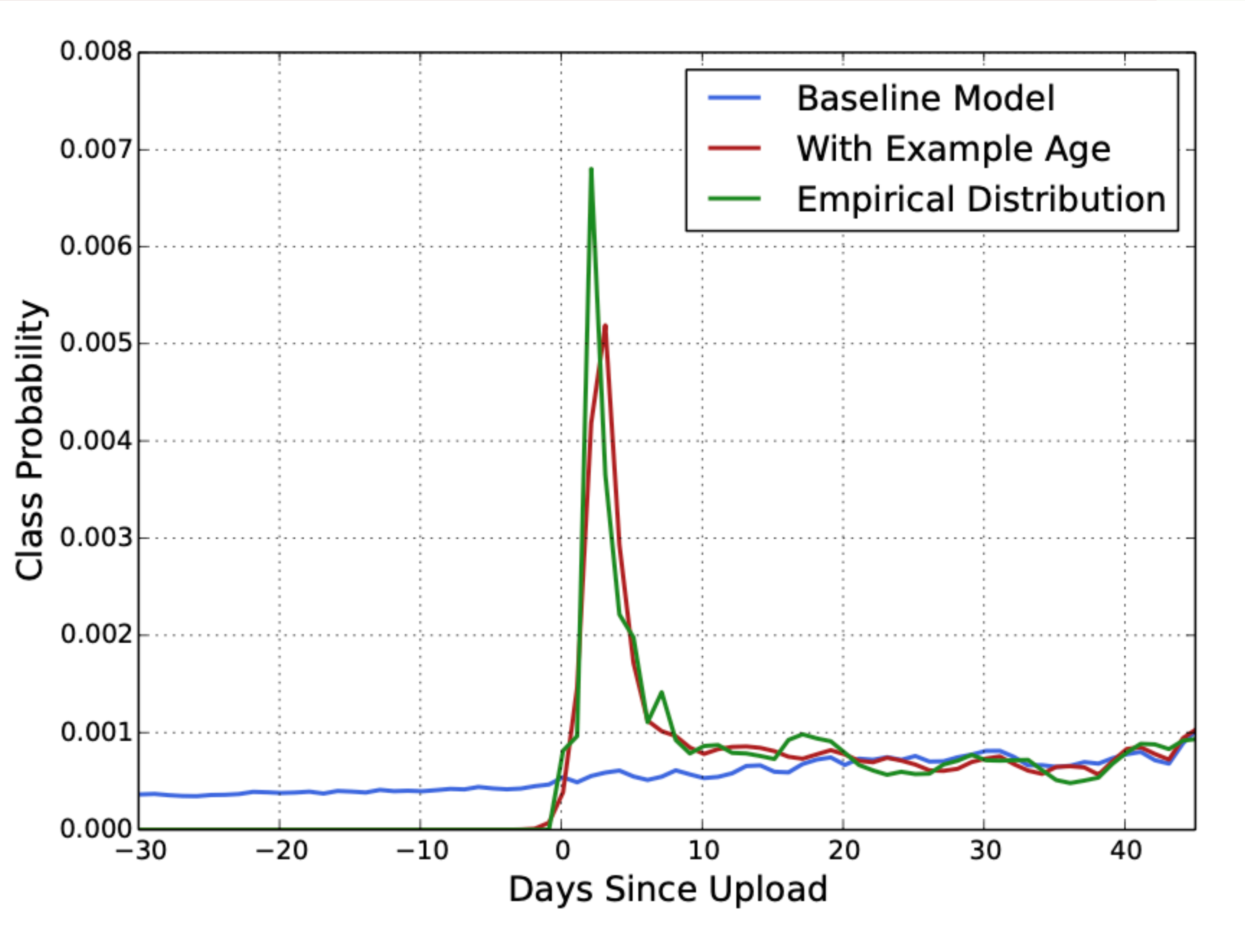
문제 top N videos 추출하기 위해 많은 계산 $\xrightarrow{\text{but}}$ scoring 해야하는 serving latency time은 정해져있음

softmax result likelihood로 scoring하면 비슷한 것을 잘 못찾음

해결책! hashing ⟶ nearest neighbor class search

Candidate Generation

Heterogenous Signals



Demographic Feature => 새로운 고객 유치에 중요

user’s 위치, 기기 -> embedding
user’s gender/log-in state/age -> real value로 input

Example age Feature

fresh videos 중요(새로운거보고싶으니까) 그것의 viral 유무 파악도 중요(영상 추천에 어려움 주기때문)
training 할때 example’s age도 feature로 포함. 학습 마지막에 영향 주도록 0이나 negative로

Candidate Generation

Model Architecture

watch history

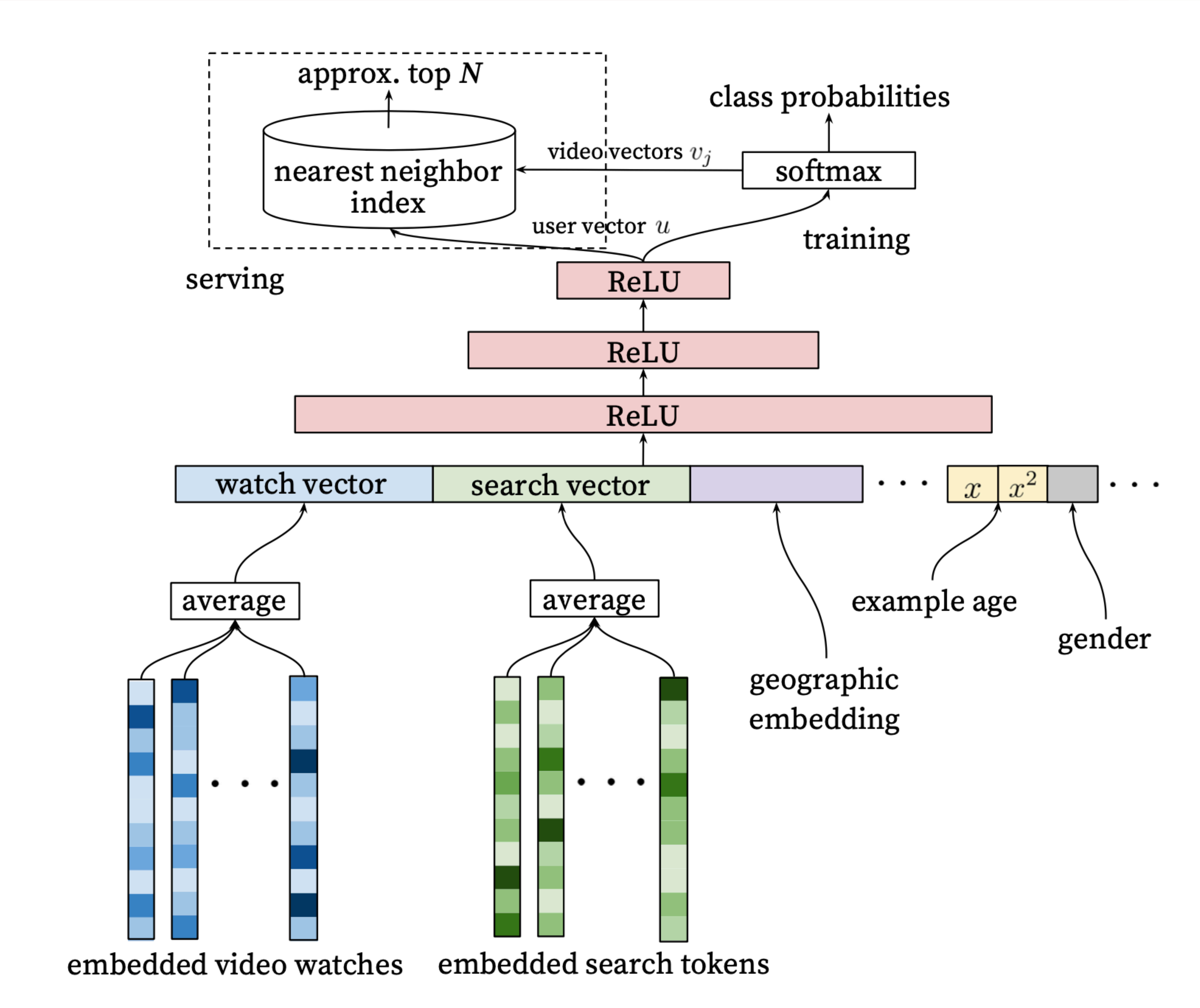
embeded해서 average vector

network는 정해진 크기의 input을 필요로 함 (다양한 길이x)

search history

watch history와 비슷

query ⟶ unigram/bigram token ⟶ embedded ⟶ vector
average



Candidate Generation

Label and Context Selection

IMPORTANT
→ Transferring classes to a particular context
→ Solving surrogate problem(in A/B testing) <간소화된 대리 데이터들에 의해 정확도 하락>

Training example = all YouTube watches | Training example fixed per user
추천 결과 시청 + 다른 사이트에서의 시청 + 등등
→ highly active user들의 취향이 bias & 추천에 의한 추천 막기 위해

~~user's last search history~~ ⟶ search quries with unordered bag of tokens
순서X
+
average vector

Candidate Generation

Label and Context Selection

<과거 데이터를 사용>

~~radom data sampling~~

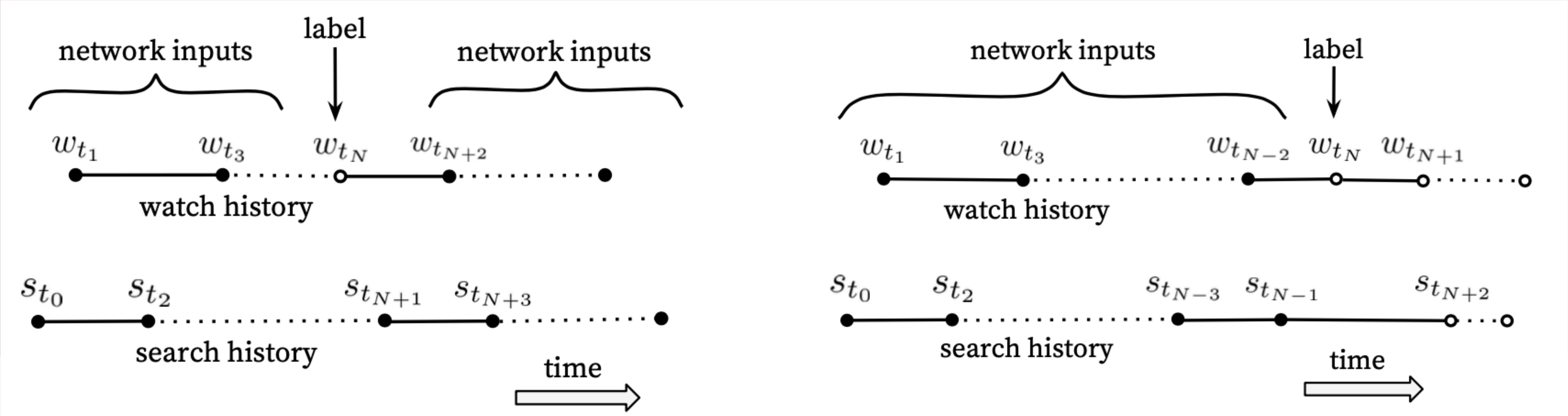
user’s history를 rollback해서 sampling

CF : randomly hold-out data로 label/context 정함

영상 시청 패턴 규칙적X , 의미없음

=> only input actions the user took

ex) series videos, videos about 특정 artist 연속적으로 볼 때
그냥 아무거나 보고 싶은 거 볼 때



(a) Predicting held-out watch (b) Predicting future watch

Candidate Generation

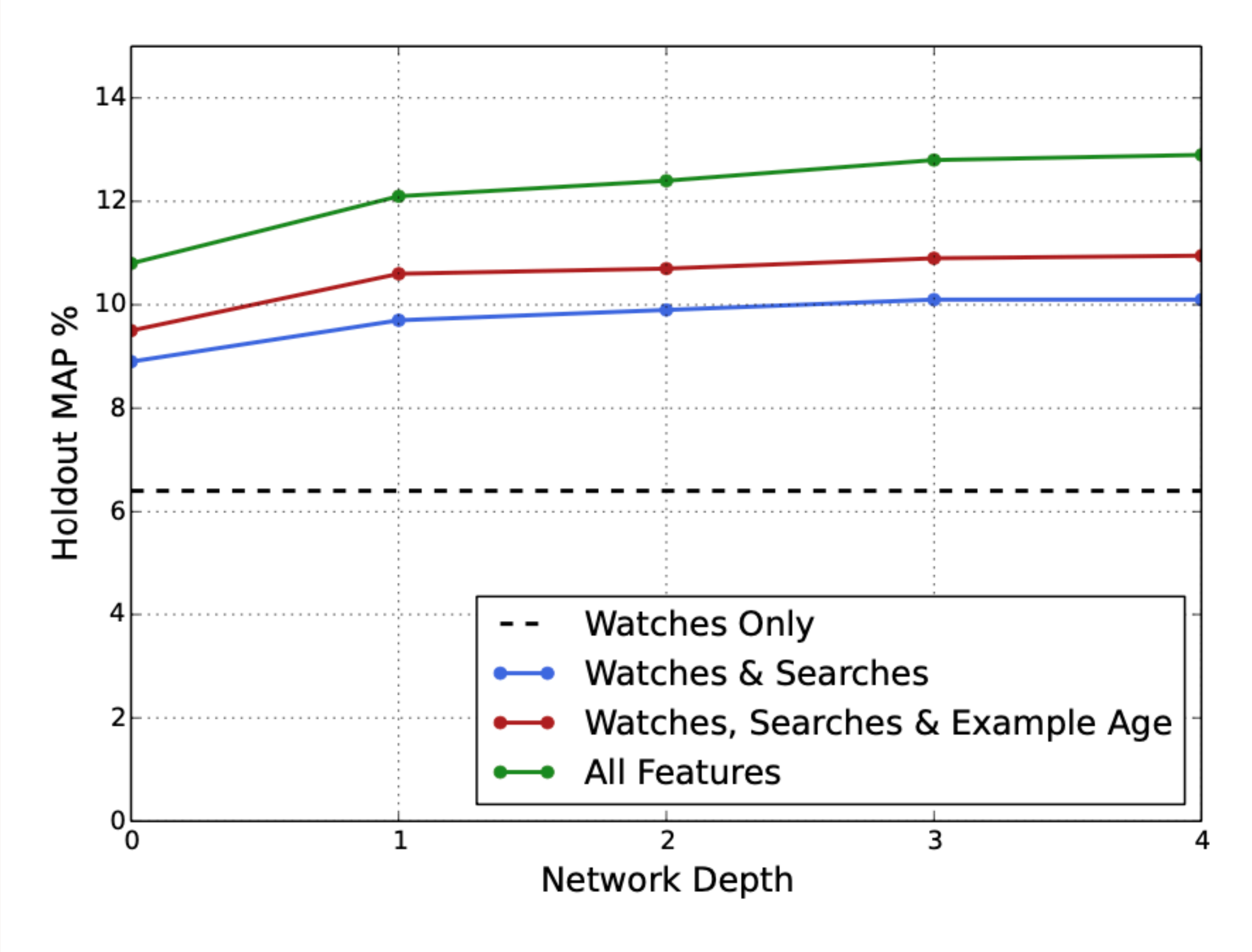
Experiments with Features and depth

(video, search token) 1M개, 256float로 embed

 -> softmax 결과가 256 float로 1M class이기 때문

수렴할 때까지 feature/depth 추가

많은 Feature + 깊은 network depth -> 정확도 높 & 수렴



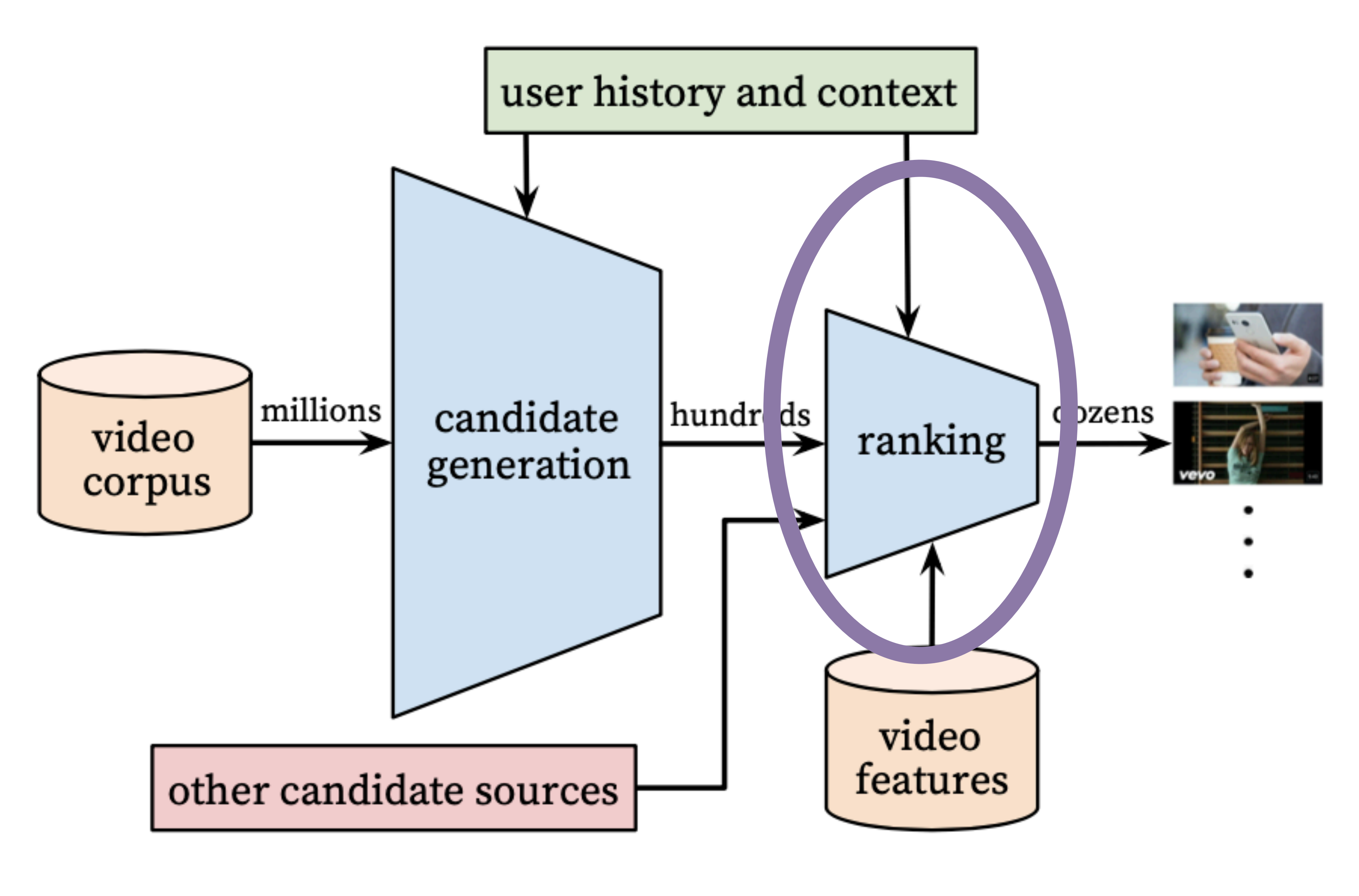
Candidate Generation

- Recommendation as classification
- Model Architecture
- Heterogeneous Signals
- Label and Context Selection
- Experiments with Features and Depth

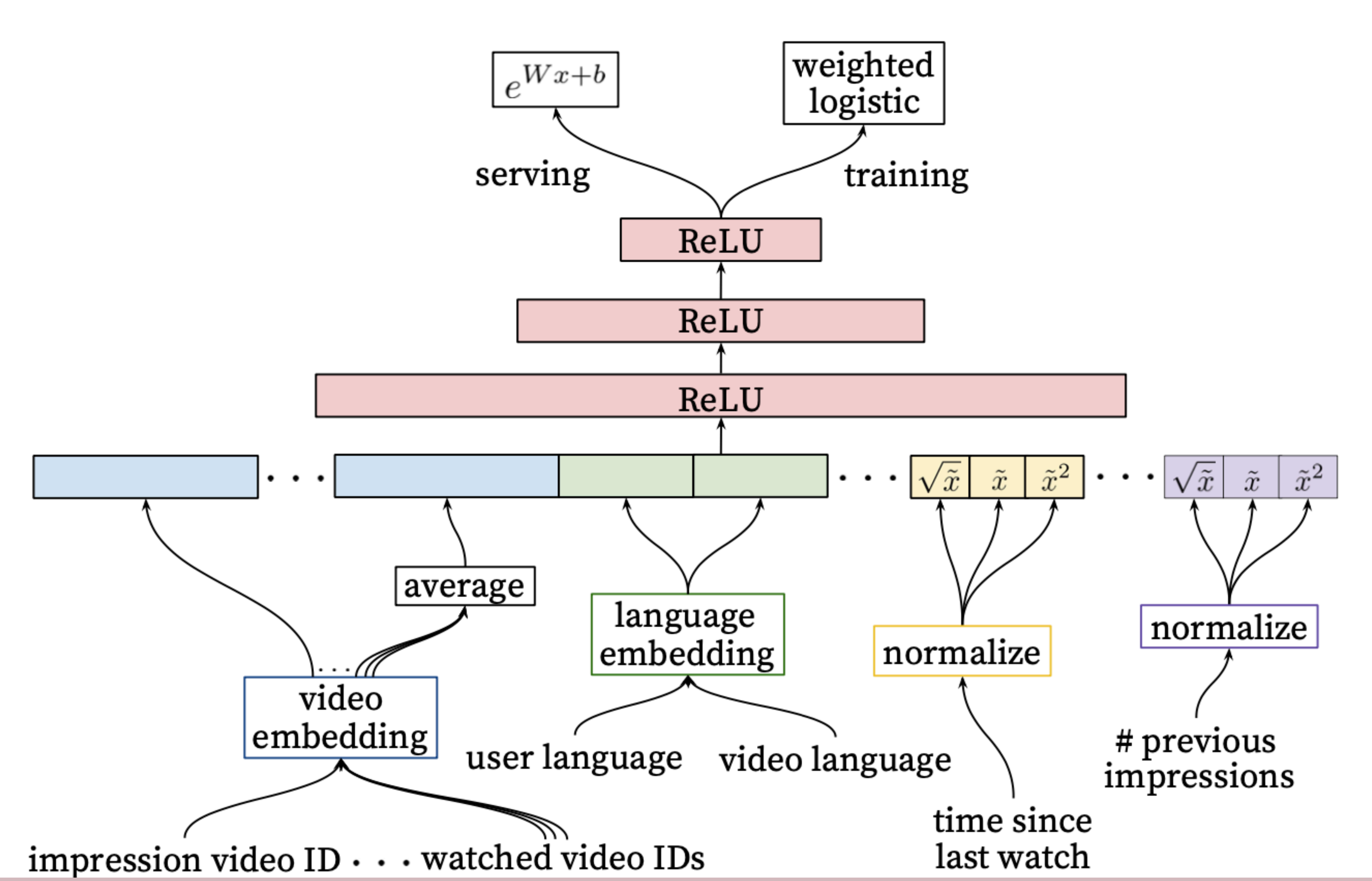
Ranking

- Feature Representation
- Modeling Expected Watch Time
- Experiments with Hidden Layers

Ranking



Ranking standard : score of expected watch time



candidate data $\xrightarrow[\text{by logistic regression + DL}]{\text{계속 교정\&전문화}}$ impression data $\xrightarrow[\text{simple function}]{}$ expected watch time about impression

live A/B test result constantly tunes ranking

Ranking

Feature Representation

Categorical Feature

Continuous Feature

여기서, 또 영향을 미친 value의 양에 따라 분류됨

single value(univalent) $\xrightarrow{\text{ex.}}$ scored impression video ID(categorical)

set of value(multivalent) → bag of the last N video IDs watched by user(continuous)

---

impression : property of item(video) // item이 scored때 마다 계산

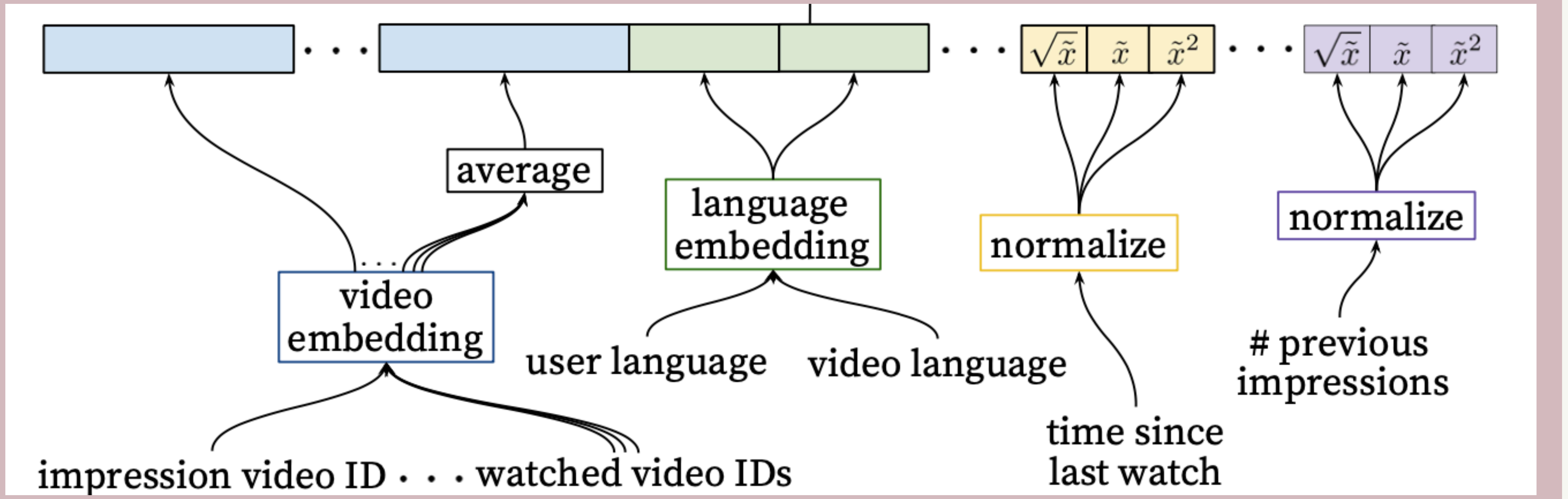
query : property of user/context // 요청에 따라 계산

Ranking

Feature Representation

Feature Enginnering

직접 engineering한 data들 DL에 input



IMPORTANT

→ user들의 행동 패턴 파악 & 그 행동과 scored video data 어떻게 연관되어있는지
영상 <->사용자, 다른 비슷한 영상<->사용자 상호작용
어떤 source로 영상이 ranking에 포함되었는지

ex)
유저가 특정 채널에서 얼마나 많은 영상을 봤는지, 유저가 특정 토픽의 동영상을 본 지 얼마나 지났는지, 영상의 과거 시청 여부

Ranking

Feature Representation

Embedding Categorical Features

unique ID space(vocabulary)

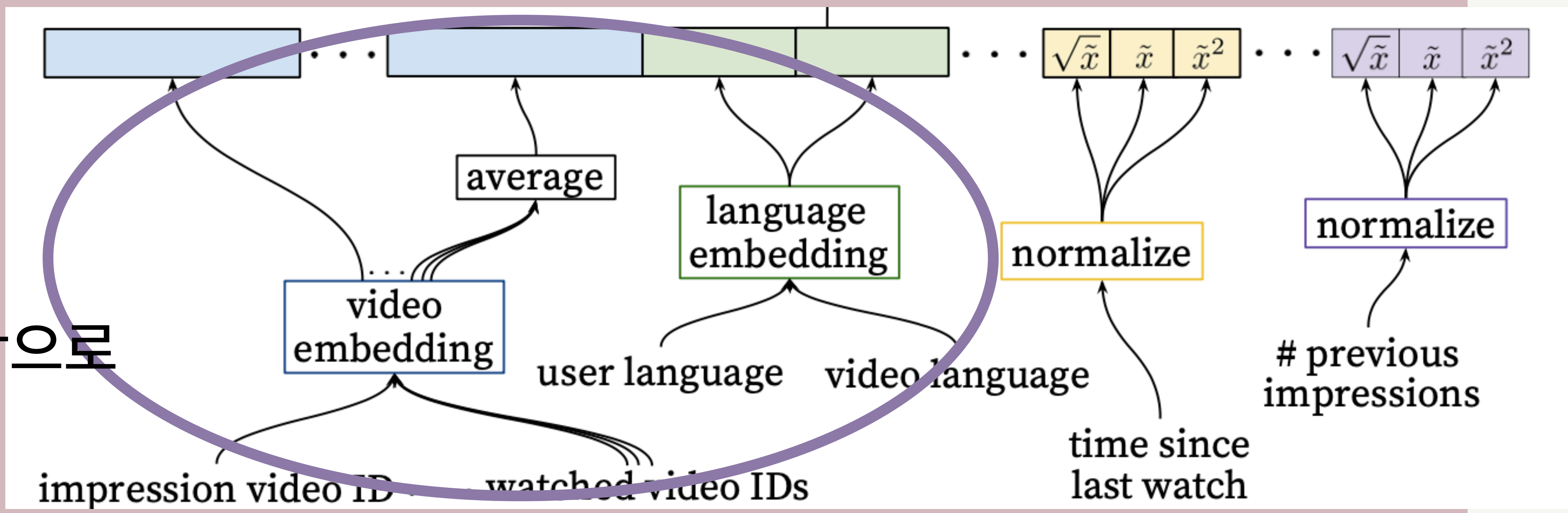
분리한 embedding 사용

large ID spaces(video IDs/search query terms)

길이를 줄임⟶클릭 빈도수로 sort해서 top N 추출함으로

top N이 아니면 zero embedding

average해서 network에 input



speed up

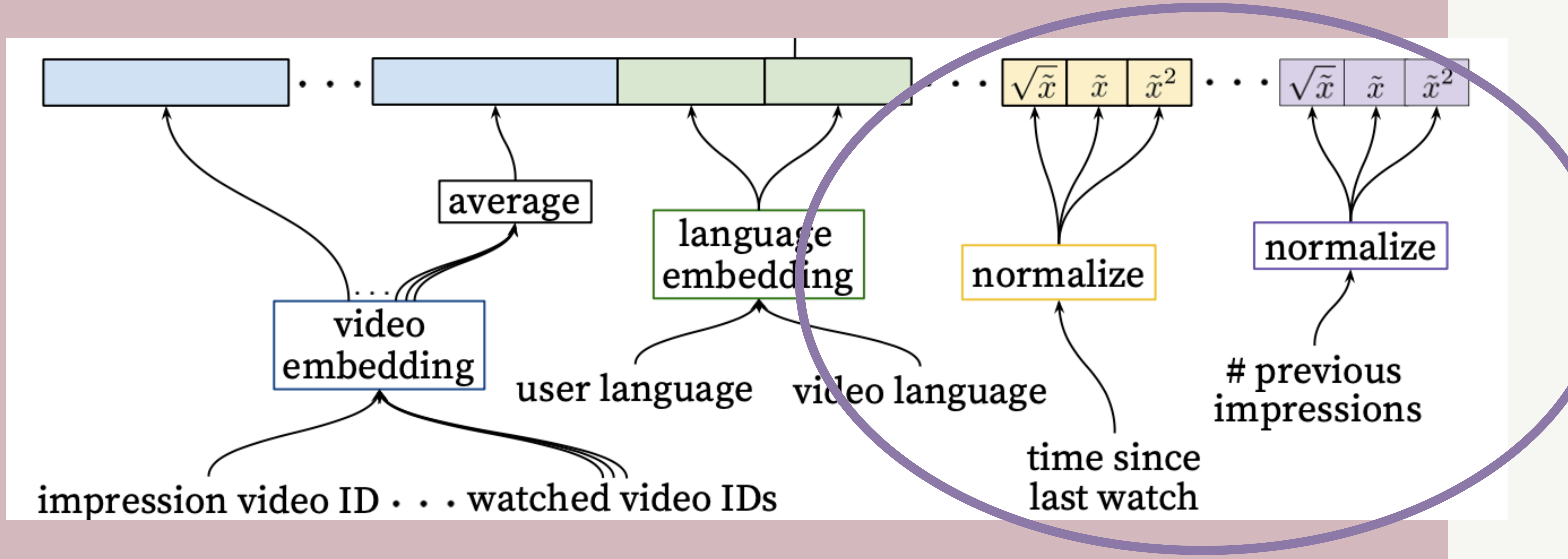
memory down

← 같은 ID space를 가진 경우 embedding 공유

but, network안에서 각각의 feature는 구분됨

Ranking

Feature Representation



Normalizing Continuous Features

적절한 scaling 필요 -> 수렴 때문에 중요행

scaling : $\tilde{x} = \int_{-\infty}^{x} \mathrm{d}f$ // f = 누적분포 함수, x = continuous feature

integral ≒ 선형 보간법

결과 값의 제곱과 루트값도 같이 넣어준다. -> 모델이 복잡한 feature들의 관계를 쉽게 학습

Ranking

Modeling Expected Watch Time

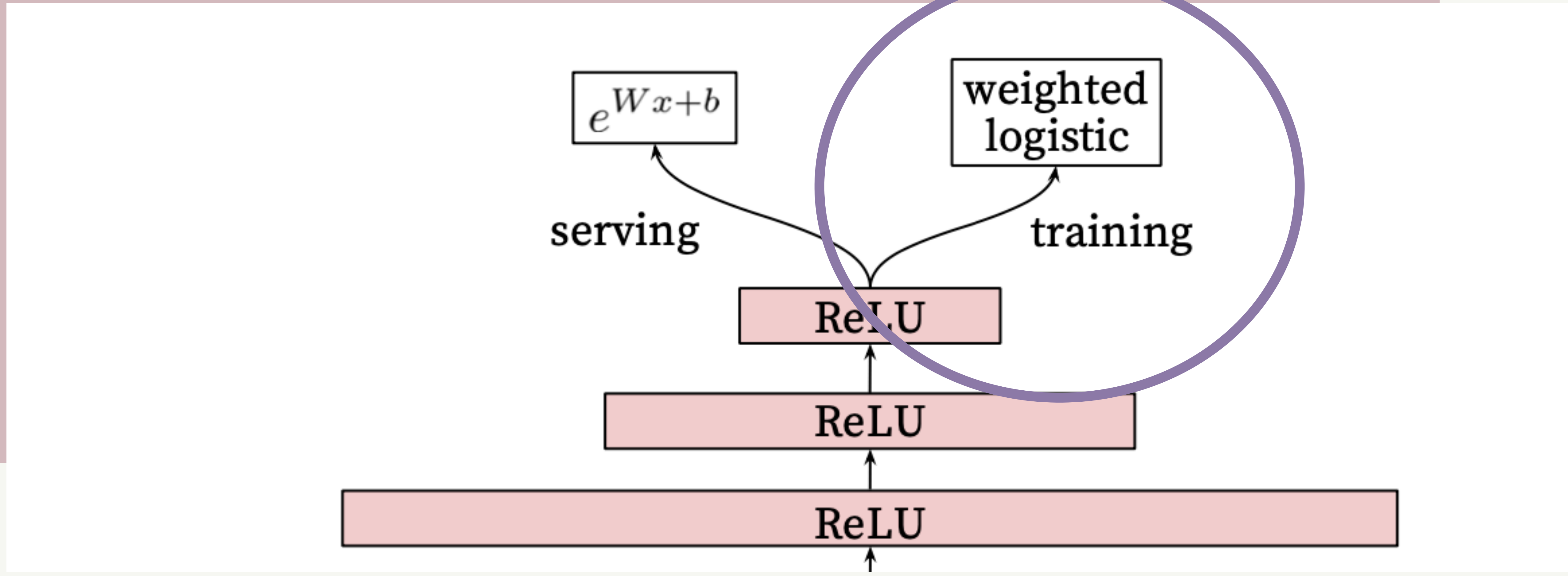
GOAL : expected watch time 예측 (이걸로 ranking하니까!)

training example - > positive(clicked) video impression: watch time 저장됨
negative(unclicked) video impression

weighted logistic regression : positive는 watch time으로 weight
negative는 unit weight

click 여부 + watch time

=> abusing



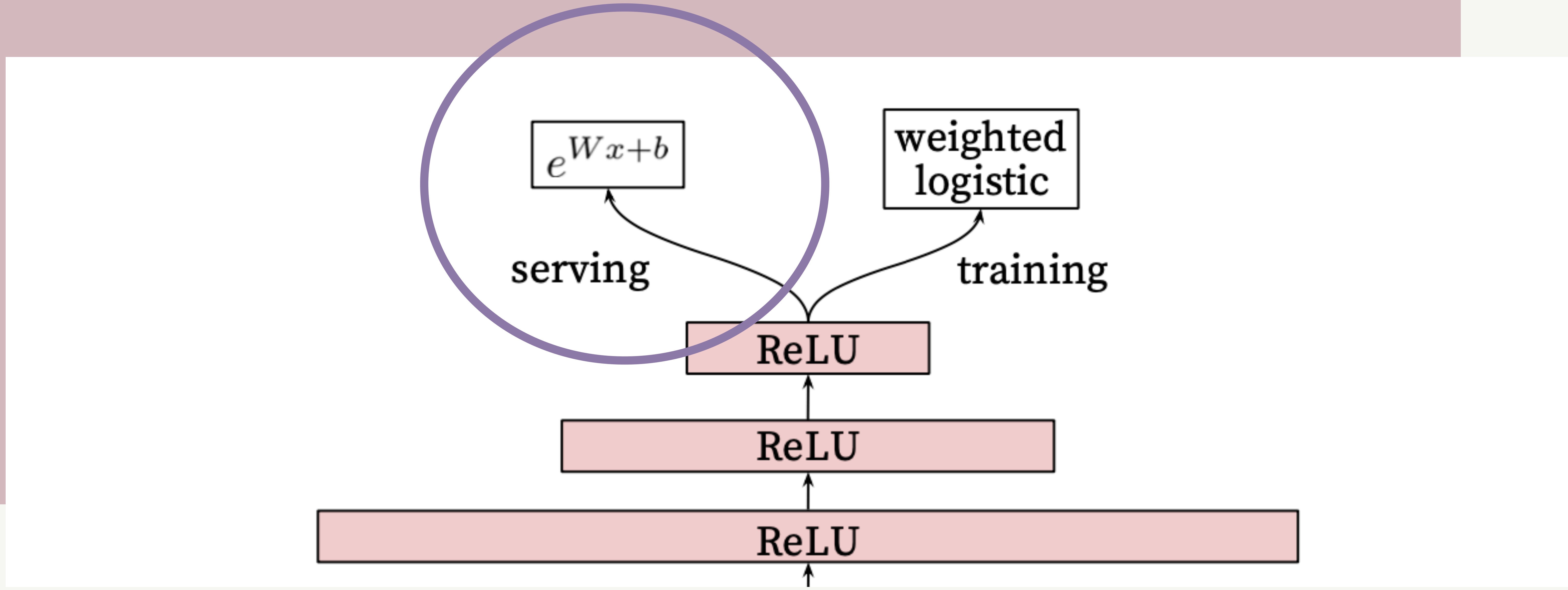
Ranking

Modeling Expected Watch Time

GOAL : expected watch time 예측 (이걸로 ranking하니까!)

serving -> odd = e^x

closely estimate expected watch time



Ranking

Experiments with Hidden Layers

Hidden layers	weighted, per-user loss
None	41.6%
256 ReLU	36.9%
512 ReLU	36.7%
1024 ReLU	35.8%
512 ReLU → 256 ReLU	35.2%
1024 ReLU → 512 ReLU	34.7%
1024 ReLU → 512 ReLU → 256 ReLU	34.6%

wider & deeper ReLu layers ⟶ lower loss