Deep Neural Networks for YouTube Recommendations

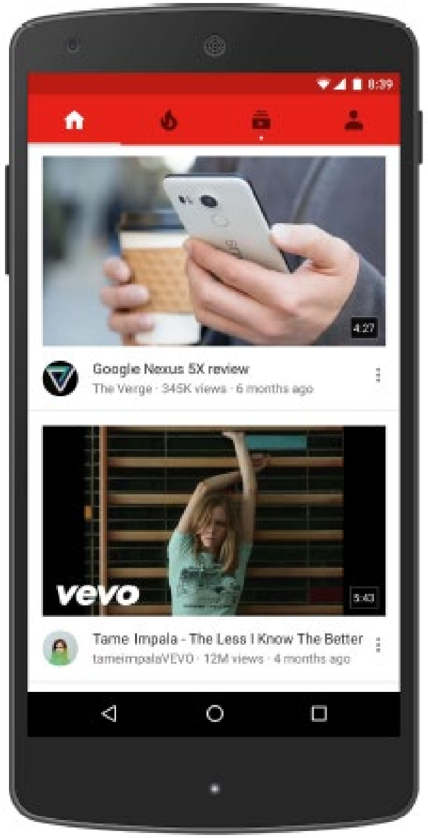
Lab Intern 이준모

bubblego0217@kaist.ac.kr

Contents

Introduction

- YouTube Recommendation
  - Responsible for helping users discover personalized content from videos
- Challenging
  - **Scale**
    - Many rec algorithms fail to operate on YouTube’s scale
    - Handling YouTube’s massive user base and corpus
  - **Freshness**
    - Be responsive enough to model newly uploaded contents, also latest actions taken by the user
  - **Noise**
    - Sparsity and a variety of unobservable external factors
    - Implicit Feedback signals



Introduction

- YouTube system
  - Fundamental paradigm shift towards using Deep Learning
  - Built on Google brain (TensorFlow : Flexible Framework)
  - Model – learn one billion parameters and are trained on hundreds of billions

- Research
  - Vast amount of research in Matrix Factorization
  - Relatively little work using DNNs for recommendations systems

Overall Structure

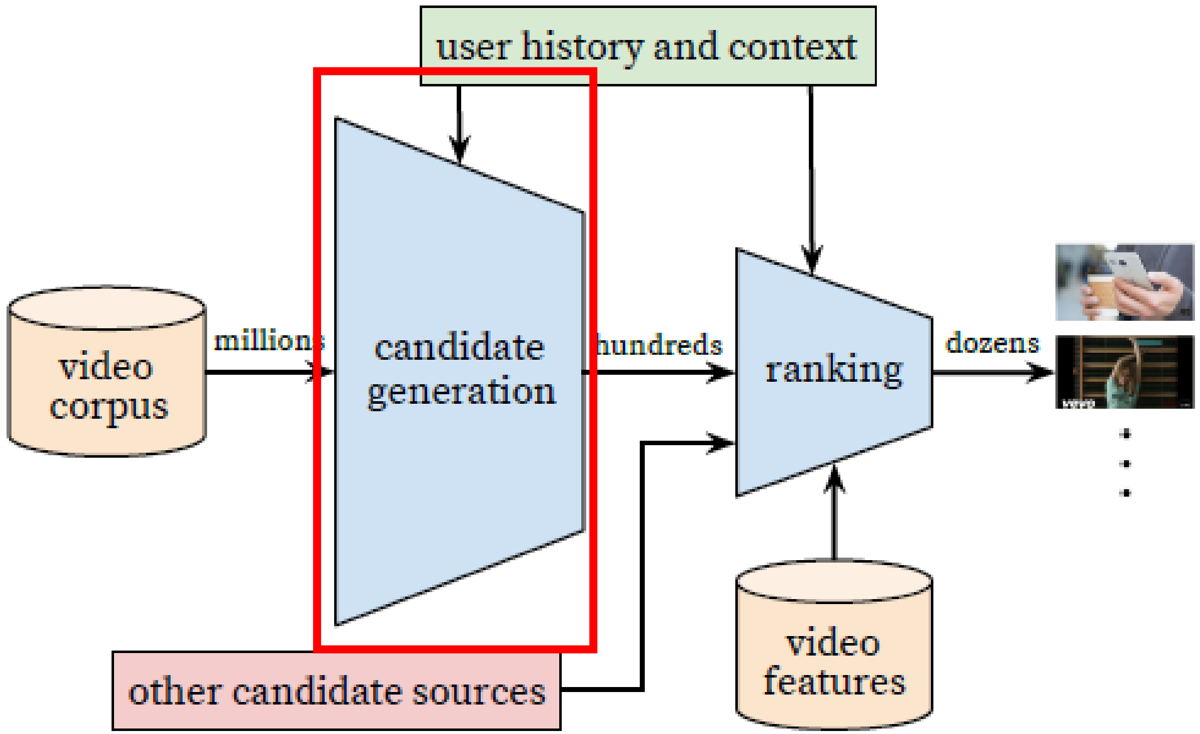


Figure 2: Recommendation system architecture demonstrating the "funnel" where candidate videos are retrieved and ranked before presenting only a few to the user.

1. Candidate Generation
   - Network takes events from user's YouTube activity history
   - Retrieves a small subset of videos from a large corpus
   - Only provides broad personalization via Collaborative Filtering
   - Similarity between users – IDs of video watches, search query tokens and demographics

Overall Structure

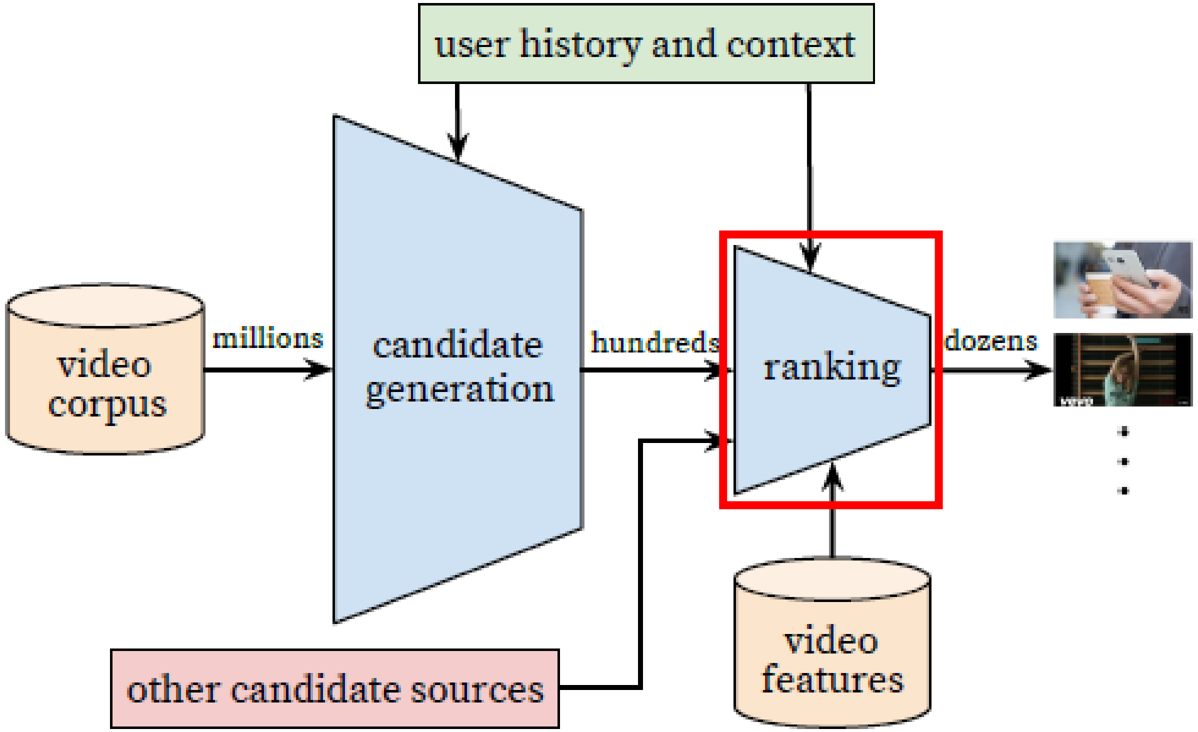


Figure 2: Recommendation system architecture demonstrating the "funnel" where candidate videos are retrieved and ranked before presenting only a few to the user.

2. Ranking

- Present a few best recommendations
  - Relative importance among candidates
- Network does by assigning a score to each video
  - Using rich set of features describing the video and user

Overall Structure

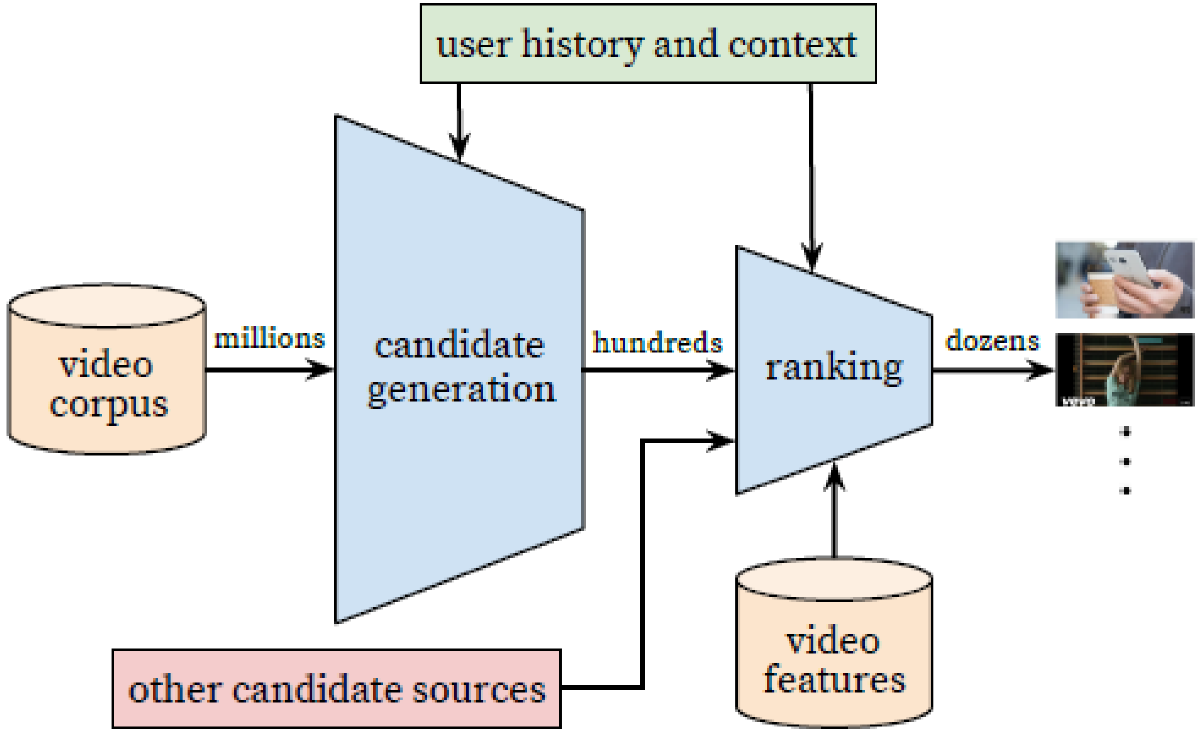


Figure 2: Recommendation system architecture demonstrating the "funnel" where candidate videos are retrieved and ranked before presenting only a few to the user.

Implication

- Can make recommendations from a very large corpus
- During development, they use offline metric (ex. Precision, Ranking loss) to improve their systems
- But they rely on A/B testing via live experiments
  - Click-through rate, Watch time

Recommendation as Classification

- Recommendation as extreme multiclass classification
  - Classifying a specific video watch w_t at time t among millions of videos i (classes) from a corpus V based on a user U and context C

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

    - $u \in R^N$: embedding of the user, context pair
    - $v_j \in R^N$: embeddings of each candidate video
  - DNN task : Learn user embeddings u
  - Use Implicit Feedback instead of Explicit Feedback
    - User completing a video is a positive example

Efficient Extreme Multiclass

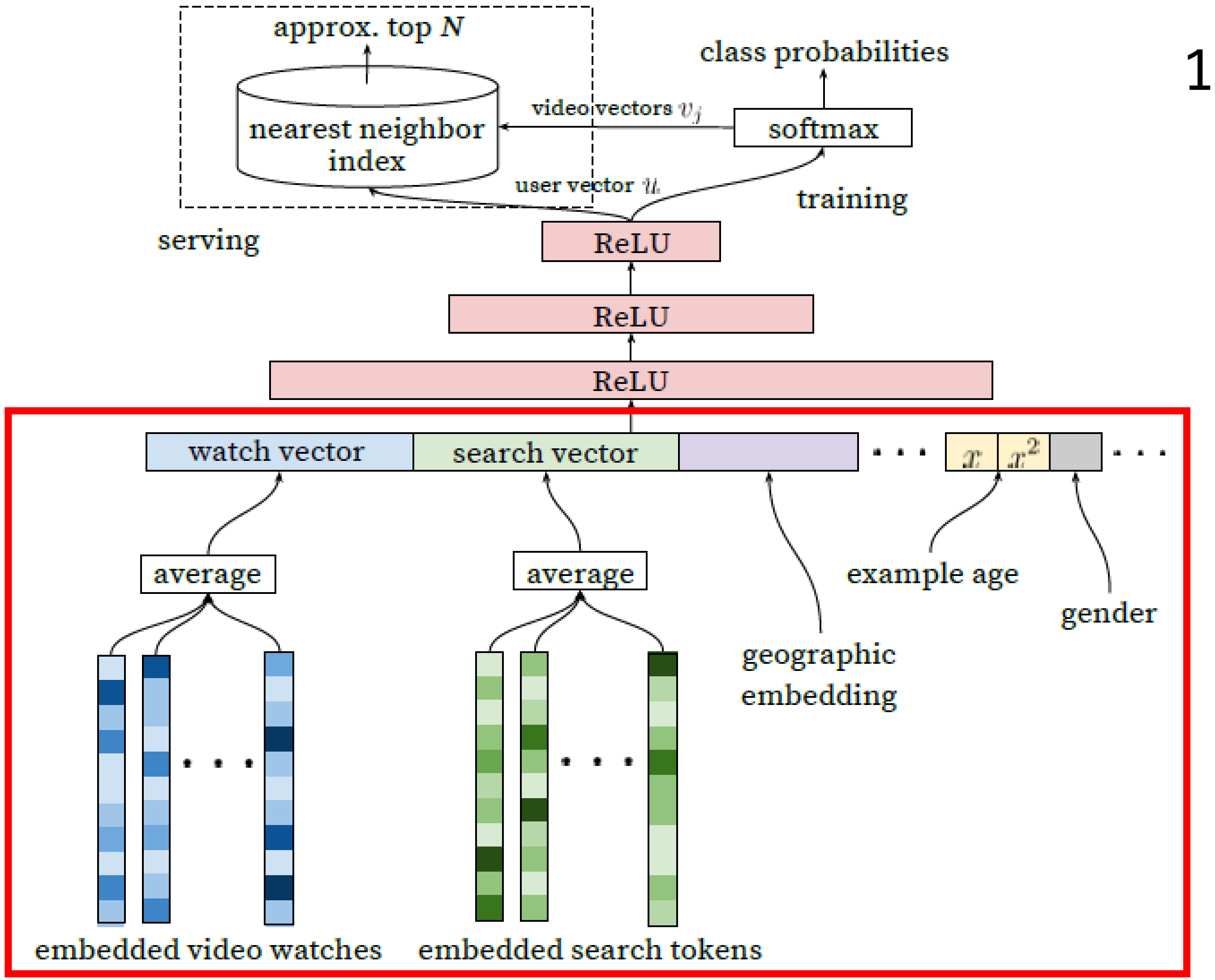
- Negative Sampling
  - To efficiently train – several thousand negatives, 100 times speedup
  - Cross-entropy loss is minimized for true label and sampled negative classes

- Serving Time
  - Compute the most likely N videos to choose the top N
    - from millions of items under serving latency of tens of milliseconds
  - The problem of extracting top N classes in multiclass classification

    = kNN search on output vector space

    - Trade off between accuracy and speed

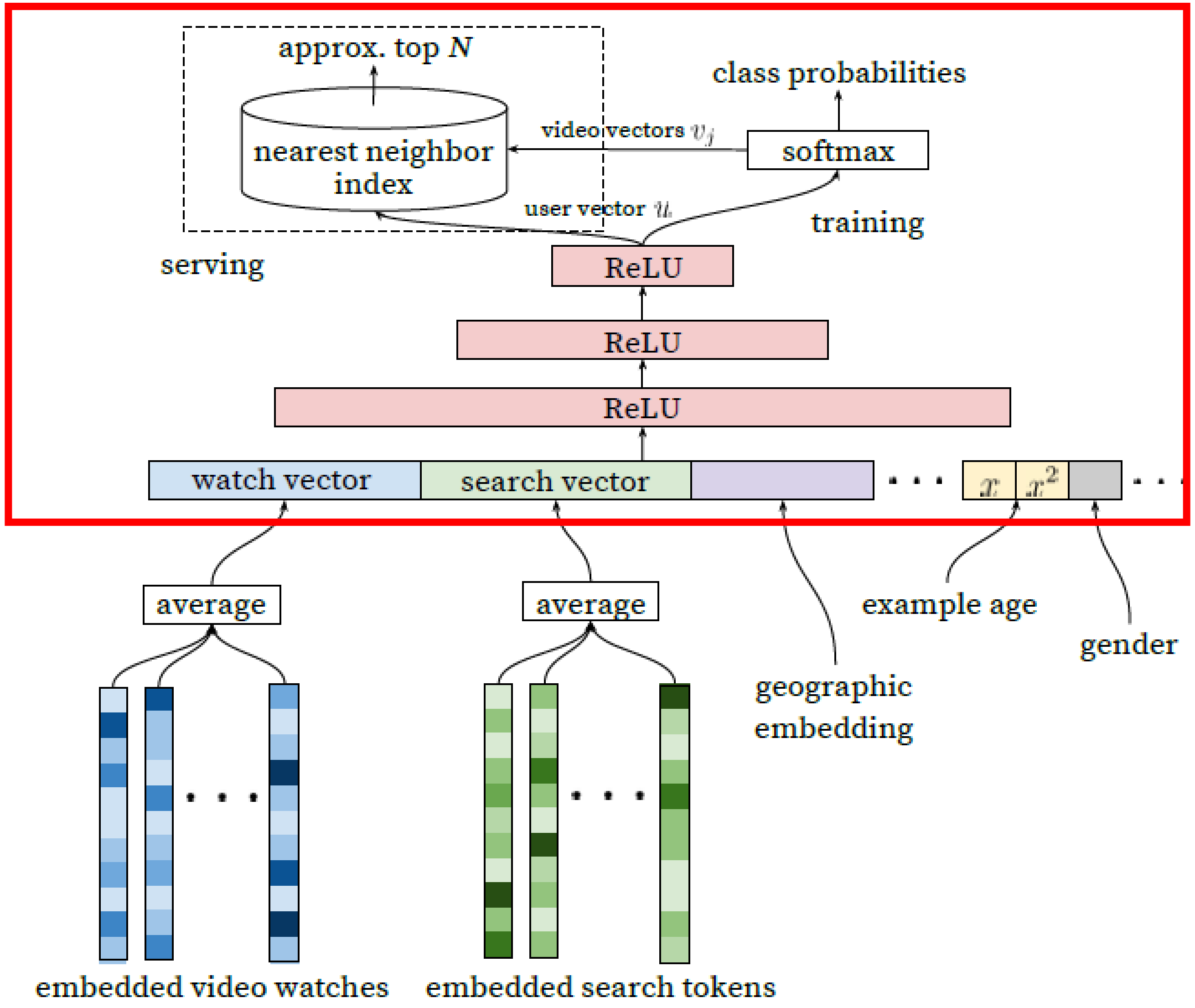
Model Architecture



1. Input Structure

- Fixed-length vector embedding for 1) video watches, 2) search keywords, 3) Demographic feature
  - Embedding averaging and aggregate
    - To avoid learning by memorizing only the last search history
    - Consider the context of the keywords user searched for in the past

Model Architecture



2. After Input Structure

- Features are concatenated into a wide first layer
- Followed by several layers of Fully Connected ReLU
- Normal gradient descent backpropagation updates

Heterogenous Signals

- Advantage of DNNs
  - Arbitrary continuous and categorical features can be easily added to model
    - Demographic features are important for reasonability
- "Example Age" Feature
  - User prefer fresh contents
  - ML often exhibits bias towards the past
    - Training from historical examples
  - Feed the age of the training example
    - Time Factor

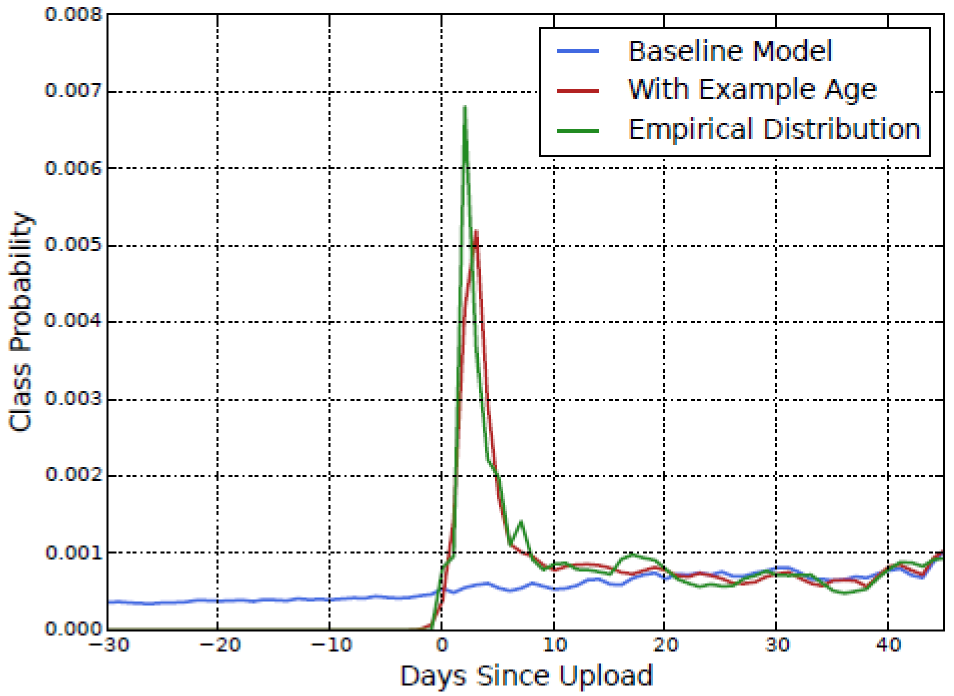


Figure 4: For a given video [26], the model trained with example age as a feature is able to accurately represent the upload time and time-dependant popularity observed in the data. Without the feature, the model would predict approximately the average likelihood over the training window.

Label and Context Selection

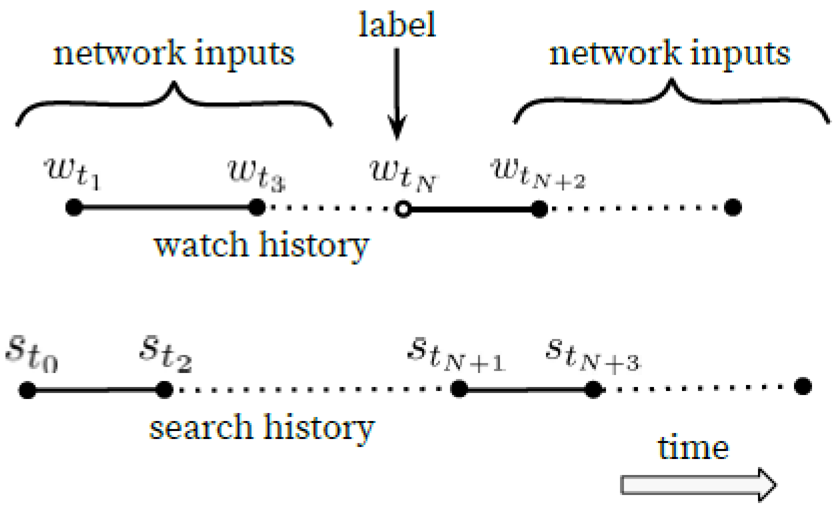
- Prepare for
  - Label – Whether the user has watched or not
  - Context – Input for candidate generation network
- Training examples
  - Generated from all YouTube watches rather than just watches on the recommendations
    - Otherwise, it would be difficult for new contents and biased
- Fixed number of training examples per user
  - Effectively weighting users equally in loss function
    - Prevent small group of highly active users from dominating the loss

Label and Context Selection

- Natural consumption patterns
  - Very asymmetric co-watch probabilities
    - Ex) Episodic series are watched sequentially

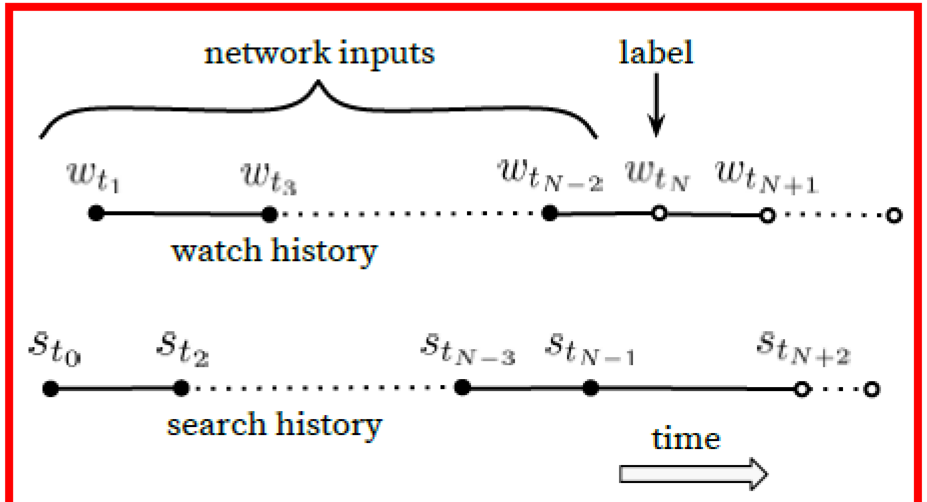
Predict the user's next watch rather than a randomly held-out watch

Many other CF systems



(a) Predicting held-out watch

YouTube Systems



(b) Predicting future watch

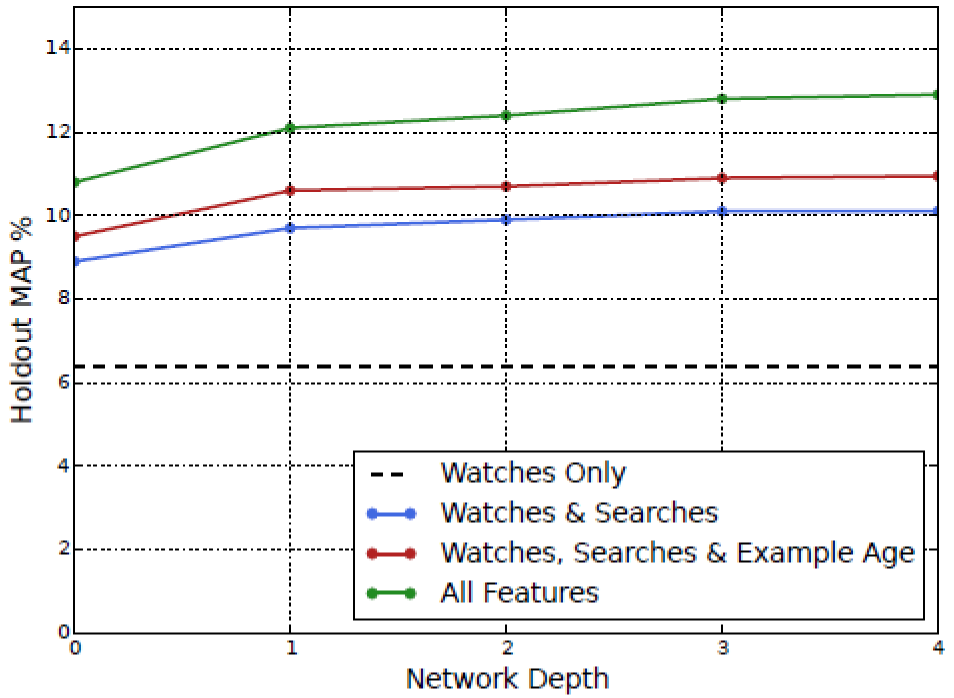
● Input features
○ excluded

Experiments with Features and Depth

- Experiment
  - 1M videos and 1M search token
  - Softmax layer output dimension : 256
  - “Tower” pattern

- Adding features and depth significantly improves precisions

- Depth 0: A linear layer simply transforms the concatenation layer to match the softmax dimension of 256
- Depth 1: 256 ReLU
- Depth 2: 512 ReLU $\rightarrow$ 256 ReLU
- Depth 3: 1024 ReLU $\rightarrow$ 512 ReLU $\rightarrow$ 256 ReLU
- Depth 4: 2048 ReLU $\rightarrow$ 1024 ReLU $\rightarrow$ 512 ReLU $\rightarrow$ 256 ReLU



Ranking

- Role of ranking
  - Use impression data to specialize and calibrate candidate predictions
  - Crucial for ensembling different sources whose scores are not directly comparable
- During Ranking
  - Access to many more features describing the video and user
    - Only a few hundred videos rather than candidate generation
- Architecture
  - Similar to candidate generation, but assign an independent score to each item
  - Ranking objective is being tuned based on live A/B testing
    - Function of expected watch time instead of click-through rate (clickbait)

Feature Representation

- YouTube's features
  - Data Type
    1. Categorical
       - Binary – whether the user is logged-in
       - Others – the user's last search query
       - ❖ Univalent – video ID of the impression being scored
       - ❖ Multivalent – a bag of the last N video IDs the user has watched
    2. Continuous/Ordinal
  - Data Meaning
    - Query - user/context feature
    - Impression – video feature

Feature Engineering

- Ranking Model
  - Typically use hundreds of features
  - Deep learning alleviate the burden of engineering features by hand
    - Raw data does not easily lend itself to be input directly
- Main Challenge
  1. Representing a temporal sequence of user actions
  2. How these actions relate to the video impression being scored

  ∵ Those are the most important signals
    - Number of videos user watched from this channel / Last time the user watched a video on this topic
      - These continuous features describing past user actions on related items

Embedding Categorical Features

- Embeddings
  - Map sparse categorical features to dense representations
  - Very large cardinality ID spaces are truncated by top N extracting based on their frequency in click
    - Video IDs, search query terms
  - Out-of-vocabulary values are simply mapped to the zero embedding
  - Multivalent categorical feature embeddings are averaged before being fed in to Network

Normalizing Continuous Features

- Normalization
  - Neural Networks are sensitive to the scaling and distribution of their inputs

→ Proper Normalization

- A feature x with distribution f is transformed to $\tilde{\mathrm{x}}$ by $\tilde{\mathrm{x}} = \int_{-\infty}^{\mathrm{x}} \mathrm{df}$
  - Integral is approximated with linear interpolation before training begins
- Also input $\tilde{\mathrm{x}}^2$ and $\sqrt{\tilde{\mathrm{x}}}$
  - Giving the network more expressive power

Modeling Expected Watch Time

- Goal
  - Predict expected watch time given training examples that are either positive (clicked) or negative
    - Positive examples have the amount of time the user spent watching the video
- Model
  - Weighted logistic regression under cross-entropy loss
    - Positive – weighted by the observed watch time
    - Negative – all receive unit weight

$$\frac{P(y=1|X;\boldsymbol{\beta})}{P(y=0|X;\boldsymbol{\beta})} = \frac{P(y=1|X;\boldsymbol{\beta})}{1-P(y=1|X;\boldsymbol{\beta})} = e^{X_i\boldsymbol{\beta}}$$

odds $\in (\mathbf{0}, \infty)$

  - The odds learned by the logistic regression $\frac{\sum T_i}{N-k}$,
    - N : num of training examples / k : num of positive impressions/ T_i : the watch time of ith impression

Modeling Expected Watch Time

- Learned odds
  - Assuming the fraction of positive impressions is small
  - Approximately $\mathrm{E[T]}(1+P)$
    - P : click probability, $E[T]$: expected watch time of the impression
    - If P is small, $\mathrm{E[T]}(1+P) \approx \mathrm{E[T]}$
  - Use the $\mathrm{e^x}$ as the final activation function to produce these odds

Deep Ranking Network Architecture

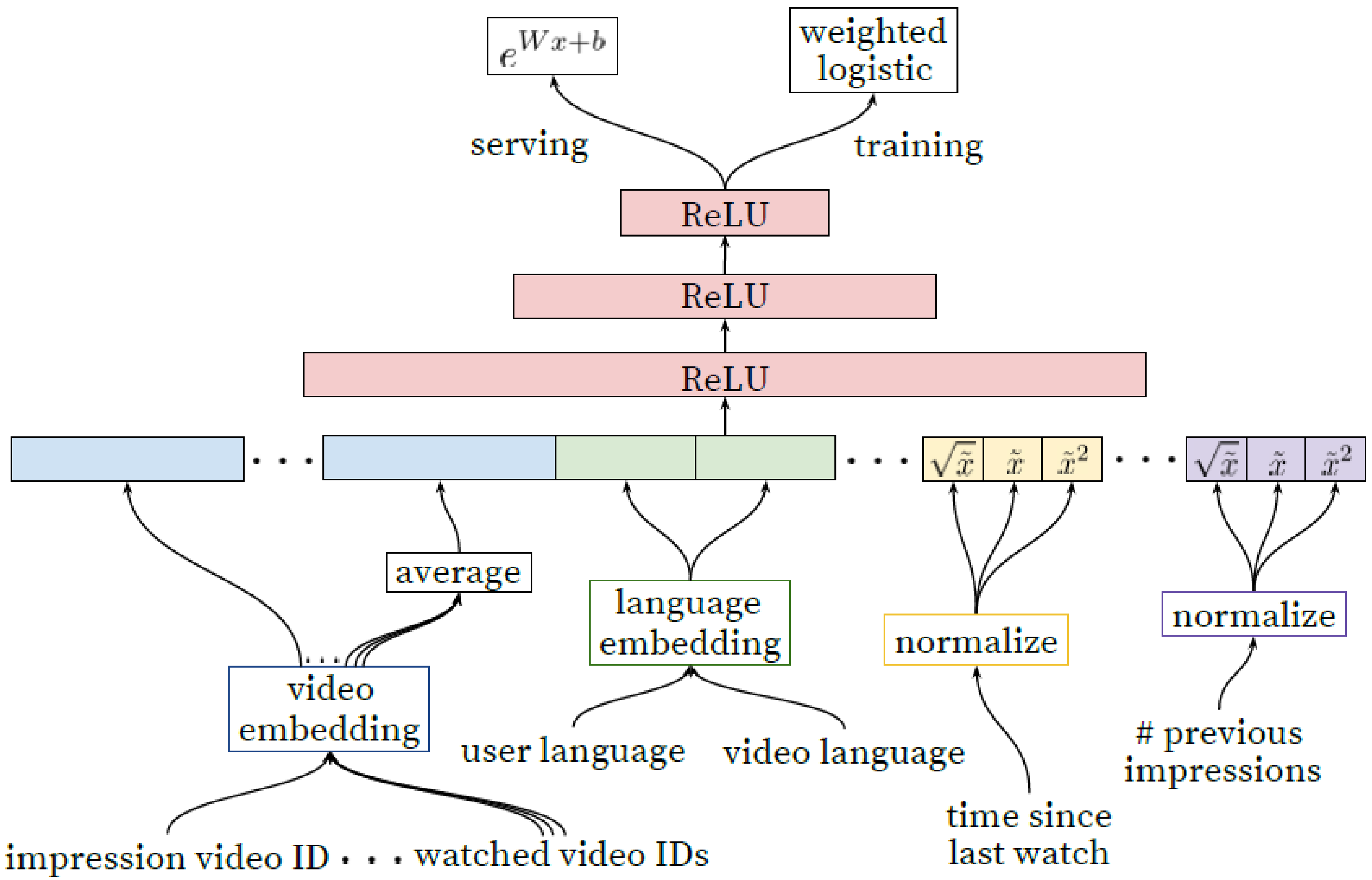


Figure 7: Deep ranking network architecture depicting embedded categorical features (both univalent and multivalent) with shared embeddings and powers of normalized continuous features. All layers are fully connected. In practice, hundreds of features are fed into the network.

Experiments with Hidden Layers

- Next-day holdout data
  - Considering both positive and negative impressions shown to a user on a single page
  - First, score these two impressions
    - Negative > Positive
      - Positive's impression's watch time to be mispredicted watch time
  - Weighted, per user loss
    - Total amount mispredicted watch time as a fraction of total watch time
- Increasing the width and depth of hidden layers improves results

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%

Table 1: Effects of wider and deeper hidden ReLU layers on watch time-weighted pairwise loss computed on next-day holdout data.

Conclusions

- DNN architecture for recommending
  - Candidate generation and Ranking
  - Assimilate many signals and model their interactions with layers of depth
- Age of the training example
  - Removes an inherent bias towards the past
  - Allows the model to represent the time-dependent behavior
- Ranking & Logistic regression
  - More classical ML problem yet, but outperformed previous linear and tree-based
  - Rec systems benefit from features describing past user behavior with items
  - Weighted logistic regression performed much better on watch time weighted ranking

Any Questions?