Deep Neural Networks for Youtube Recommendations

GROUP 26
Prashant Ramnani (17CS10038)
Kousshik Raj (17CS30022)
Satyam Porwal (17CS10048)

CONTENTS

1

Introduction and Motivation

YouTube

- World’s largest platform for creating, sharing and discovering video content
- Youtube recommendations are responsible for helping more than billion users to discover personalized content from an ever-growing corpus of videos

Challenges in YouTube Recommendations

FRESHNESS

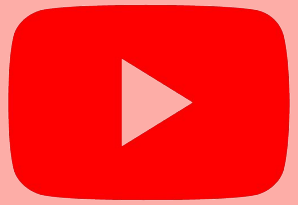
YouTube has a very dynamic corpus with many hours of video uploaded per second and thus the recommendation system has to be highly responsive.

SCALE

Existing recommendation algorithms fail to operate on the large scale. Highly specialised distributed learning algorithm and efficient service are needed to handle massive user base and corpus.

NOISE

Historical user behavior on YouTube is inherently difficult to predict due to sparsity and a variety of unobservable external factors.



YouTube

Youtube has undergone a paradigm shift towards deep learning as general purpose solution for nearly all learning problem

Previous works were based on matrix factorization methods. The deep learning based work in field of recommendation was very limited.

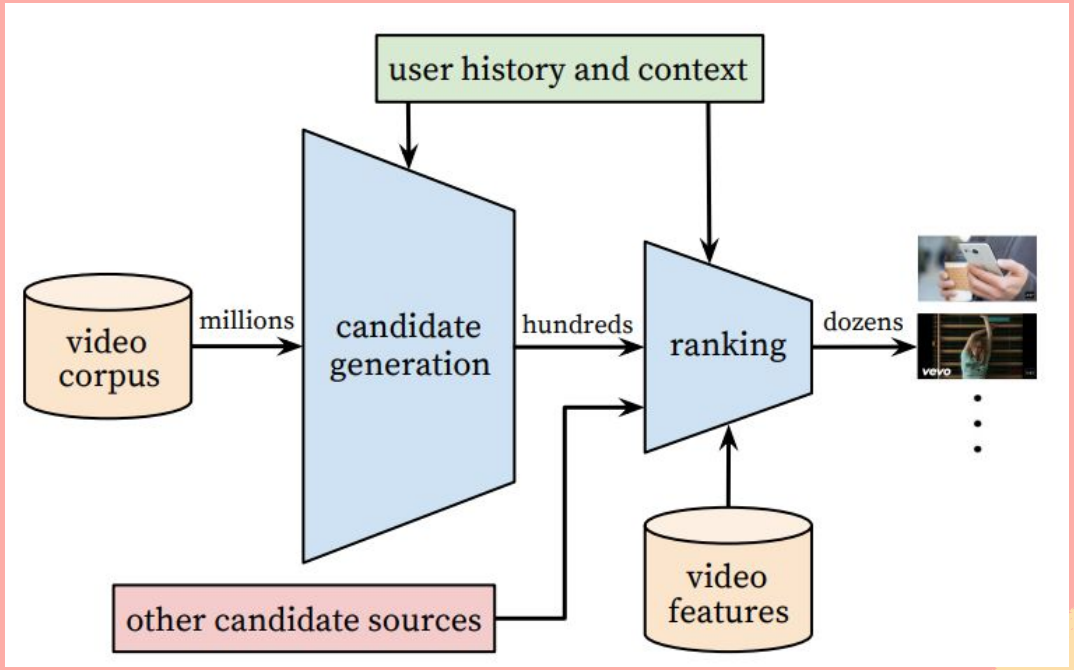
2

System Overview

SYSTEM OVERVIEW

Candidate Generation

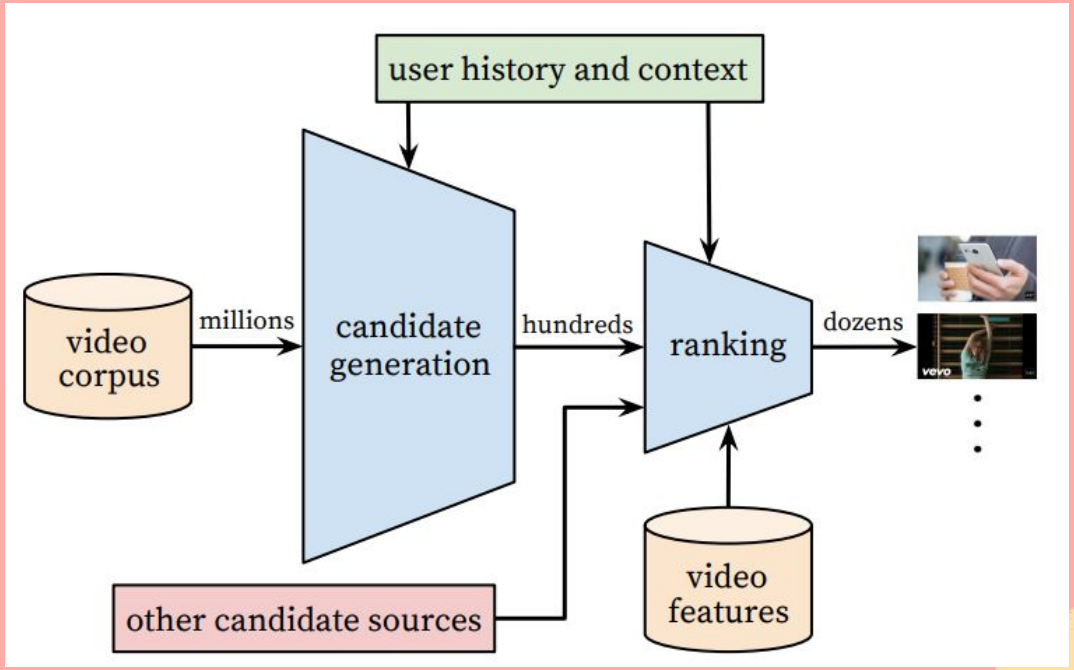
- Takes events from user's YouTube activity history and retrieves small subset of relevant videos with **high precision**
- The candidate generation network only provides broad personalization via collaborative filtering



SYSTEM OVERVIEW

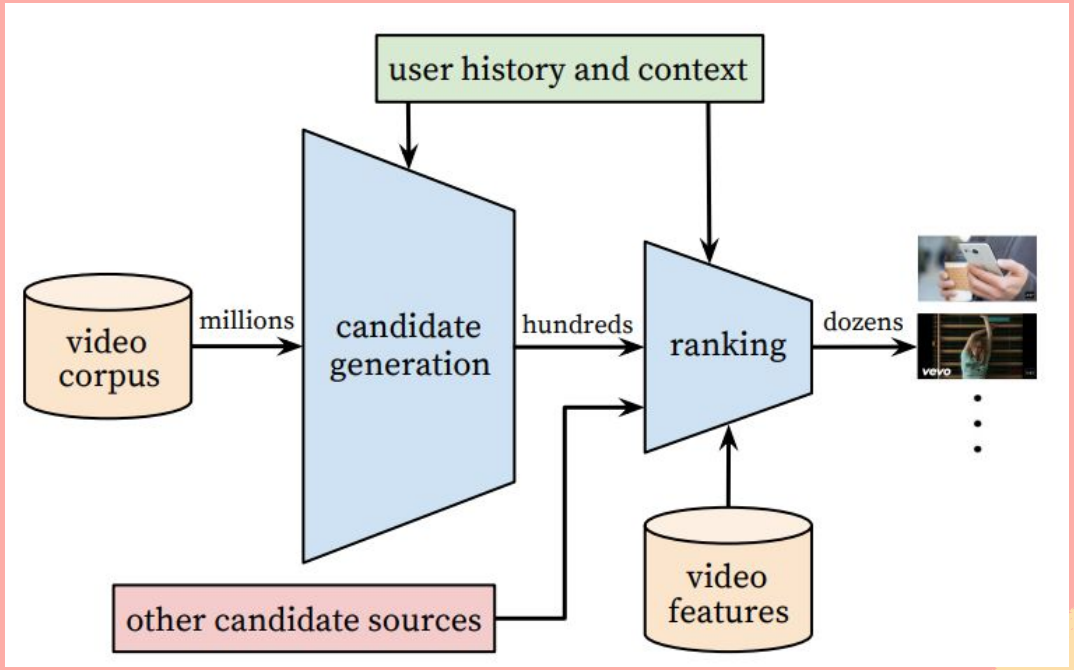
RANKING

- Presenting a few “best” recommendations among the list of candidates with **high recall**.
- The ranking network accomplishes this task by assigning a score to each video based on features describing the video and user.



SYSTEM OVERVIEW

- This makes the generated recommendations more personalized and engaging.
- Various offline metrics like precision, recall, ranking loss were used to guide iterative improvements. Live A/B testing is used for final determination of effectiveness.



3

Candidate Generation

Early Works

- The predecessor to the recommender was a matrix factorization approach trained under rank loss.
- Early iterations of the candidate generation model mimicked this factorization behaviour with shallow networks that only embedded the user's previous watches.
- In this perspective, the proposed approach behaves like a nonlinear generalization of factorization technique.

Recommendation as Classification

- The recommendation system can be treated as multiclass classification problem.

- The prediction problem being classifying a specific video watch w_t at time t among millions of videos *i* (classes) from corpus V based on a user U and context C,

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

where $u \in R^N$ represents high-dimensional “embedding” of user, context pair and $v_j \in R^N$ which represents “embedding” of each candidate video

Recommendation as Classification

- The task of the neural network is to learn user embeddings **u** as a function of user's history and context which are useful for discriminating among videos with softmax classifier.

- Even though there exists explicit feedback mechanism in YouTube, the proposed model used implicit feedback like user completing a video, as positive example.

Recommendation as Classification

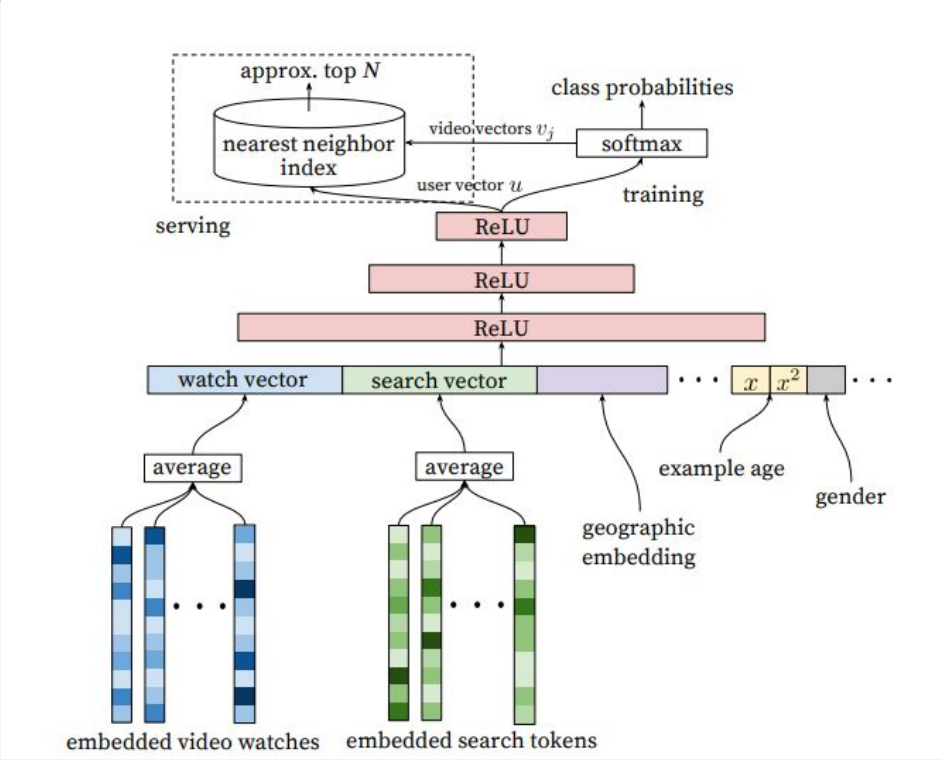
Efficient Extreme Multiclass:

- Scoring millions of items under a strict serving latency of a few milliseconds requires an approximate scoring scheme sublinear in number of classes.
- Previous system relied on hashing and classifier network.
- The proposed network used a technique to sample negative classes from background distribution (candidate sampling) and then correct for this sampling via importance weighting

Model Architecture

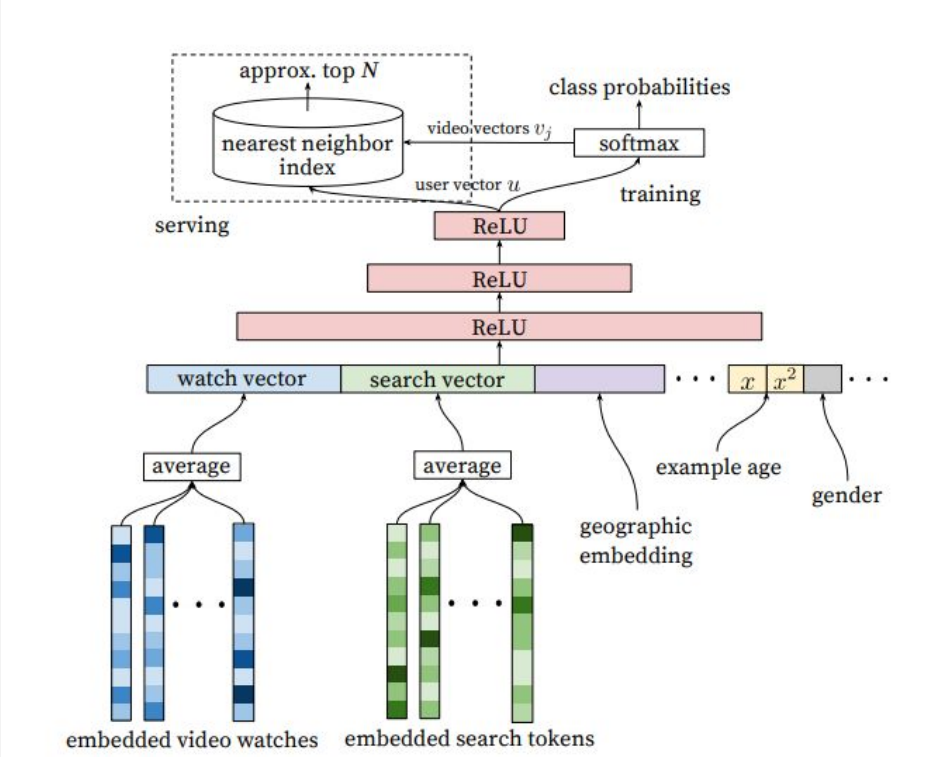
- Inspired by Continuous Bag of words, the model tries to learn high dimensional embeddings for each video in fixed vocabulary and feed these embeddings into a feedforward neural network

- A user's watch history is represented by a variable-length sequence of sparse video IDs which is mapped to a dense vector representation via the embeddings



Model Architecture

- Simply averaging the embeddings performed best among several strategies.
- The embeddings are learned jointly with all other model parameters through normal gradient descent backpropagation updates.
- Features are concatenated into a wide first layer, followed by several layers of fully connected Rectified Linear Units (ReLU).



Heterogeneous Signals

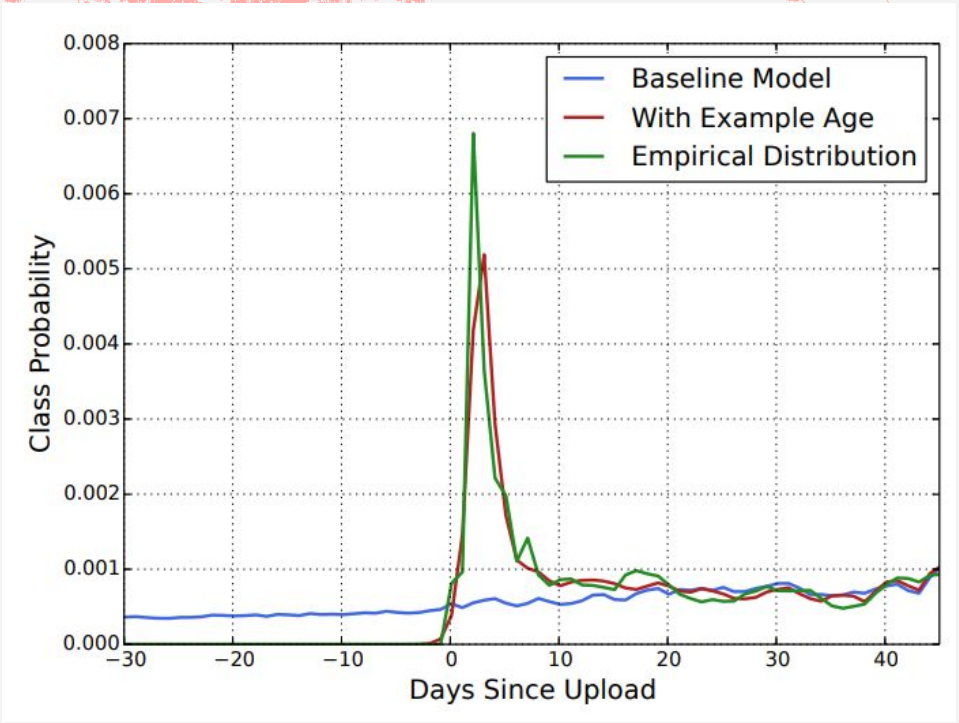
- An advantage of deep neural networks is that arbitrary continuous and categorical features can be easily added to the model.
- Search history is treated the same as watch history.
- Averaged user tokenized embedded queries represent summarized dense search history.

Heterogeneous Signals

- Demographic features are important for providing priors for new users. User’s geographic region and device are embedded and concatenated.
- Simple binary and continuous features such as the user’s gender and age are input directly into the network as normalized values to [0,1].
- Another important feature used is “Example Age”

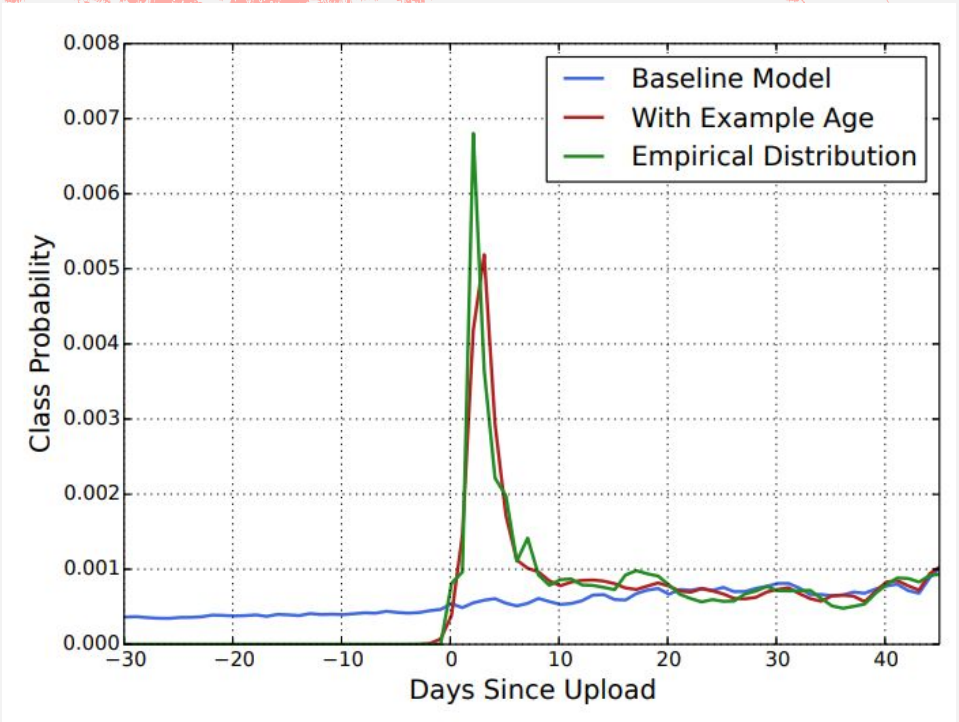
“Example Age” Feature

- Recommending fresh content is extremely important
- There’s also a critical phenomenon of bootstrapping and propagating viral content.
- Machine learning systems often exhibit an implicit bias towards the past since they are trained to predict the future based on historical examples.



“Example Age” Feature

- The distribution of video popularity is highly non-stationary.
- The multinomial distribution over the corpus produced by recommender reflects the average watch likelihood in the training window of several weeks.
- To correct this, the Age of training example (“Example Age”) is fed as a feature during training.
  And set to zero or slightly negative at serving time.



Label and Context Selection

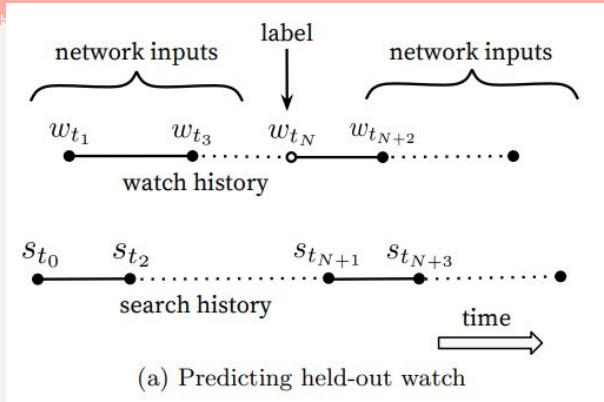
- Recommendation often involves solving a surrogate problem and transferring the result to a particular context.
- The choice of this surrogate learning problem has an outsized importance on performance in A/B testing but is very difficult to measure with offline experiments.
- Training examples are generated from all YouTube watches and not just watches on the recommendations produced. This ensures that the recommender brings new content to surface and is not overly biased towards exploitation

Label and Context Selection

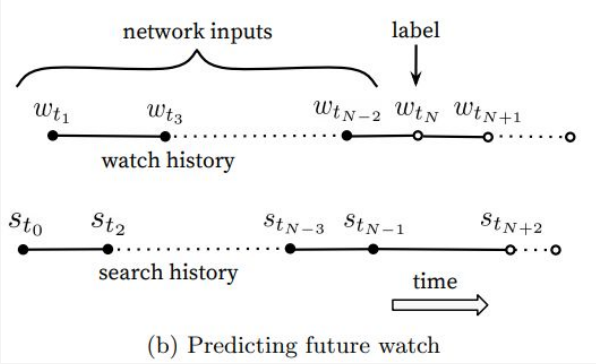
- If a user discovers videos through means other than recommendations the discovery should be quickly propagated via collaborative filtering.
- It is important to generate a fixed number of training examples per user, effectively weighting the users equally in the loss function. This prevents highly active users from dominating the loss.
- It is important to withhold information from the classifier in order to prevent the model from exploiting the structure of the site and overfit the surrogate problem.

Label and Context Selection

- Reproducing the user's last search page as homepage recommendations performs very poorly.
- ***Natural consumption patterns*** of videos typically lead to very asymmetric co-watch probabilities.
- Many collaborative filtering systems implicitly choose the labels and context by holding out a random item and predicting it from other items in the user's history
- In contrast, we "rollback" a user's history by choosing a random watch and only input actions the user took before the held-out label watch



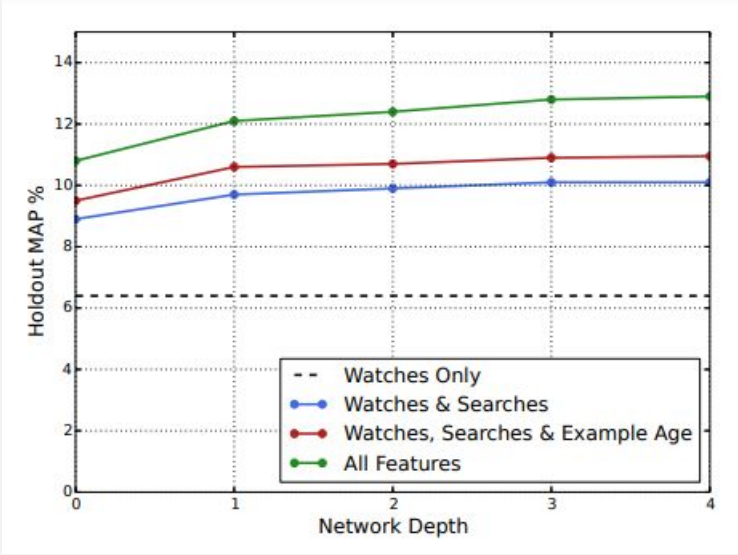
(a) Predicting held-out watch



(b) Predicting future watch

Experiments with Features and Depth

- Adding features and depth significantly improves precision on holdout data.
- The network structure followed a common “tower” pattern, the bottom layer is the widest and each successive hidden layer halves the number of units.
- The depth zero network is effectively a linear factorization scheme which performs very similar to the predecessor system.
- Features beyond video embeddings improve Holdout Mean Average Precision and layers of depth add expressiveness so that the model can effectively use these additional features by modeling their interaction.



4

Ranking Network

The Idea

- Uses impression data to specialize and calibrate candidate predictions for the particular user interface.
- Also ensembles different candidate sources.
- Uses deep neural network to assign independent scores to videos and returns the sorted list.
- Ranking objective is constantly tuned based on the responses from live A/B testing, usually expected watch time.

Feature Representation

- Numerous more features (in hundreds) incorporated owing to smaller number of videos to handle.
- Expend considerable resources to transform user and video data to useful features by hand.
- The challenge is to represent the temporal sequence of user actions and how they will relate to the scoring of the impressions.

What Are Some Important Features?

- Most important signals to the network are those that describe a user's previous interaction with the item itself and other similar items.
- It was also found crucial to propagate information from candidate generation into ranking.
- Features describing the frequency of past video impressions are also critical for introducing "churn" in recommendations.

Categorical Features

- These are features that generally take limited number of possible values. Moreover, the values need not be numerical. Eg: video ID, search query, etc.
- These are sparse categorical features and needed to be converted into dense representations like the embeddings used previously.
- Each unique feature space (vocabulary) has a separate learned embedding whose dimension is proportional to the logarithm of the number of unique values.
- The vocabulary is built by making a pass over the training data.

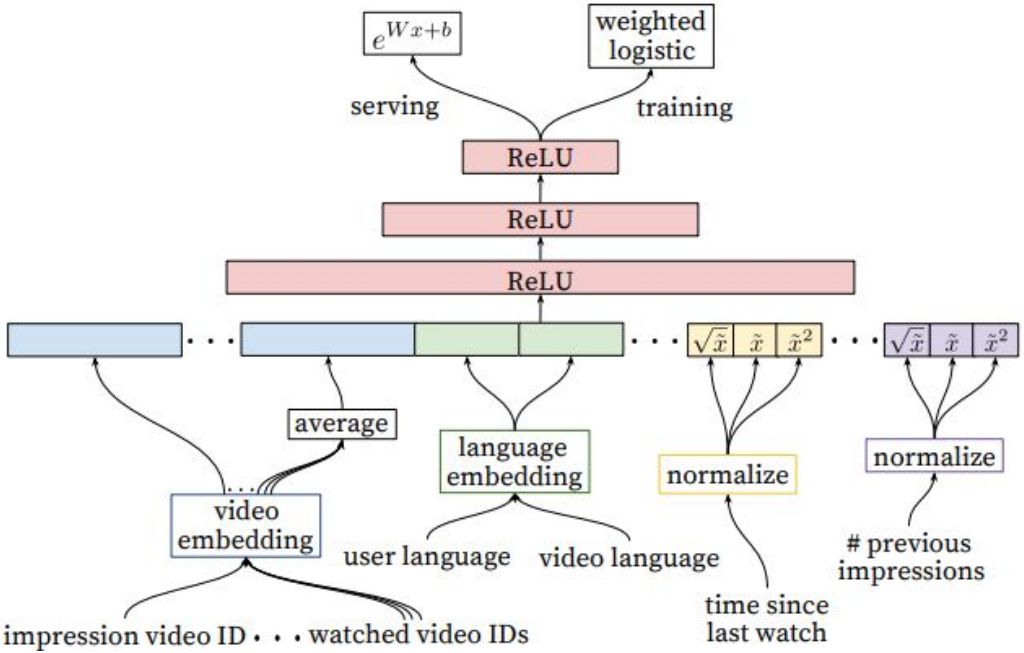
Categorical Features

- Very large cardinality feature spaces (eg:- video id) are truncated to only N most frequent entities. Out of vocabulary values are mapped to the zero embedding.
- Multivalent categorical features are averaged before input to the network.
- Features that share feature space use the same underlying embedding. Eg:- video impression ID, last watched video ID, similar video ID.
- This increases performance and improves generalisation. It also decreases memory requirements.

Continuous Features

- These features can take infinite number of possible values and are unbounded. Eg:- time since last watch, etc.
- Neural networks are extremely sensitive to the scaling and distribution of inputs and thus normalisation is needed for convergence.
- A continuous feature ***x*** with distribution ***f*** is transformed to $\tilde{x}$ by scaling the values such that the feature is equally distributed in [0, 1) using the cumulative distribution, $\tilde{x} = \int_{-\infty}^{x} \mathrm{d}f$
- Addition to the raw normalized value $\tilde{x}$, $\tilde{x}^2$ and $\sqrt{\tilde{x}}$ are also passed to the network.

Network Architecture



- Hundreds of features fed.
- Tower like layers, with each subsequent layer size halved.
- Learns to assign expected watch time to impressions based on the input features.

Model Training

- Goal is to predict expected watch time of videos, given training example impressions are of positive or negative types.
- Positive types are also annotated with the amount of time user spent watching it.
- The model uses the technique of weighted logistic regression, a modified variant of logistic regression trained under cross entropy loss, with each example weighted.

Model Training

- Here, the positive examples are weighted with their watch time and the negative examples receive unit weight.
- Thus the odds learned are $\mathbf{\sum T_i / (N - k)}$. Here, k is the number of positive examples and T_i is the watch time of the i^{th} positive impression.
- For small k, the odds boil down to E[T](1 + P), where P is the click probability.
- There is also a final exponential activation layer (the final score), and the odds of resultant scores closely resemble expected watch times.

Experiments

- The results are obtained by live A/B testing. The impressions shown in the page are segregated into positive and negative.
- The model first scores all the impressions. If a negative impression receives higher score than a positive one, then positive impression's watch time is *mispredicted watch time*.
- "**Weighted, per user loss"** is the ratio of the sum of the mispredicted watch time and the total watch time over all the impression pairs

Experiments

- Increasing width and depth of the layers improves the results, but at the cost of server CPU - time.
- When only raw normalized features without their powers were passed, loss increases by 0.2%
- When all examples are weighted equally, the loss increases by 4.1%

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%

THANK YOU