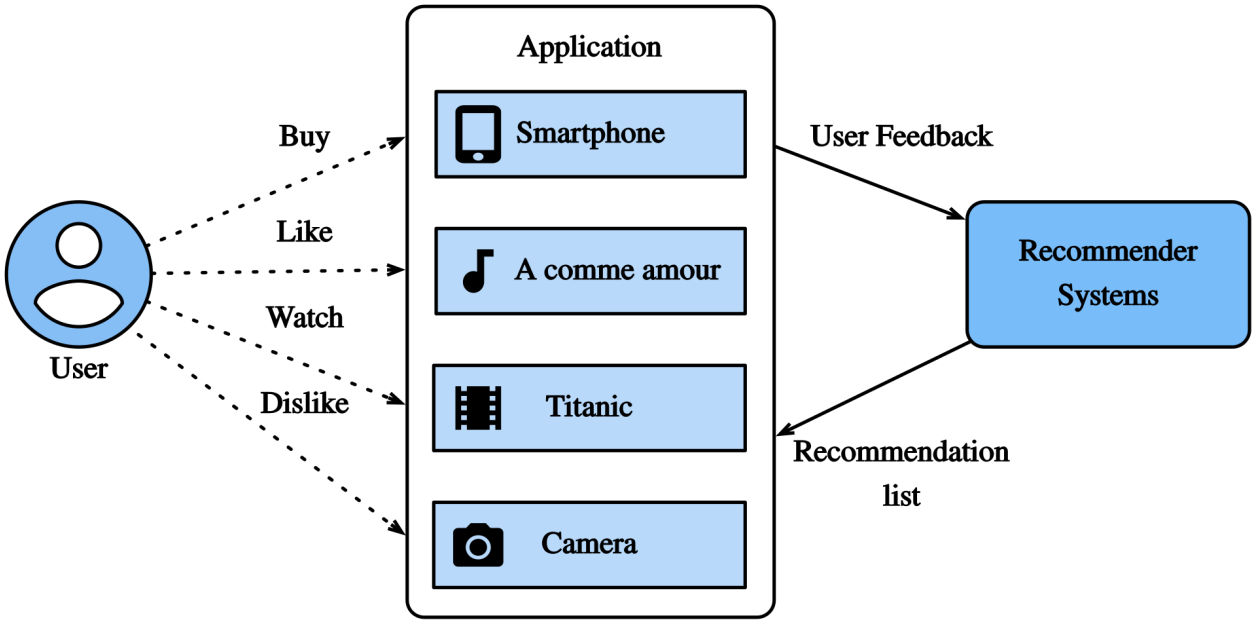
Wide & Deep Learning for Recommender Systems

Heng-Tze Cheng, Levent Koc, Jeremiah Harmsen, Tal Shaked, Tushar Chandra, Hrishi Aradhye, Glen Anderson, Greg Corrado, Wei Chai, Mustafa Ispir, Rohan Anil, Zakaria Haque, Lichan Hong, Vihan Jain, Xiaobing Liu, Hemal Shah

Kalmutskiy Kirill

April 2021

Recommendation tasks



A recommender system can be viewed as a search ranking system, where:

- the input query is a set of user and contextual information;
- the output is a ranked list of items.

Given a query, the recommendation task is to find the relevant items in a database and then rank the items based on certain objectives, such as clicks or purchases.

Memorization vs Generalization

One challenge in recommender systems, similar to the general search ranking problem, is to achieve both memorization and generalization.

- Memorization
  - learning the frequent co-occurrence of items or features and exploiting the correlation available in the historical data;
  - Ability to remember training data;
  - Tends to recommend best possible items.
- Generalization
  - based on transitivity of correlation and explores new feature combinations that have never or rarely occurred in the past;
  - Ability to learn from training data;
  - Tends to improve the diversity of the recommended items;

Memorization vs Generalization

- **Memorization**: "Seagulls can fly", but "can penguin fly?"
- **Generalization**: "Animal with wings can fly", but penguin can't!

Memorization: The Wide Component

The Wide Component is a generalized linear model $y = w^T x + b$, trained on sparse and high-rank features.

The main idea is to use cross-product transformation, which is defined as:

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

where c_{ki} is a boolean variable that is 1 if the i-th feature is part of the k-th transformation ϕ_k, and 0 otherwise. More simple, it's like new features using AND operation.

This captures the interactions between the binary features, and adds nonlinearity to the generalized linear model.

Generalization: The Deep Component

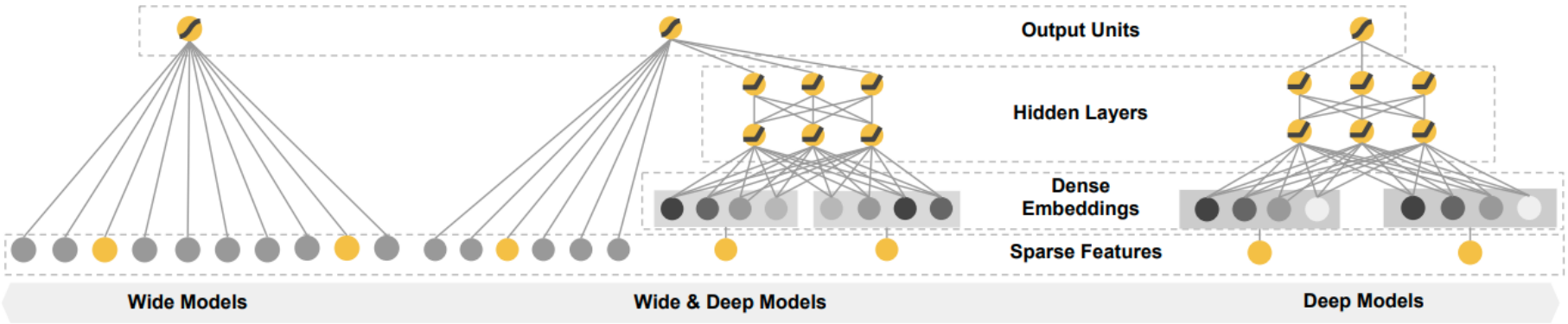
The deep component is a simple feed-forward deep neural network.

The main idea is to convert high-dimensional categorical features into a low-dimensional and dense real-valued vector, often referred to as an embedding vector.

These low-dimensional dense embedding vectors are then fed into the hidden layers of a neural network in the forward pass.

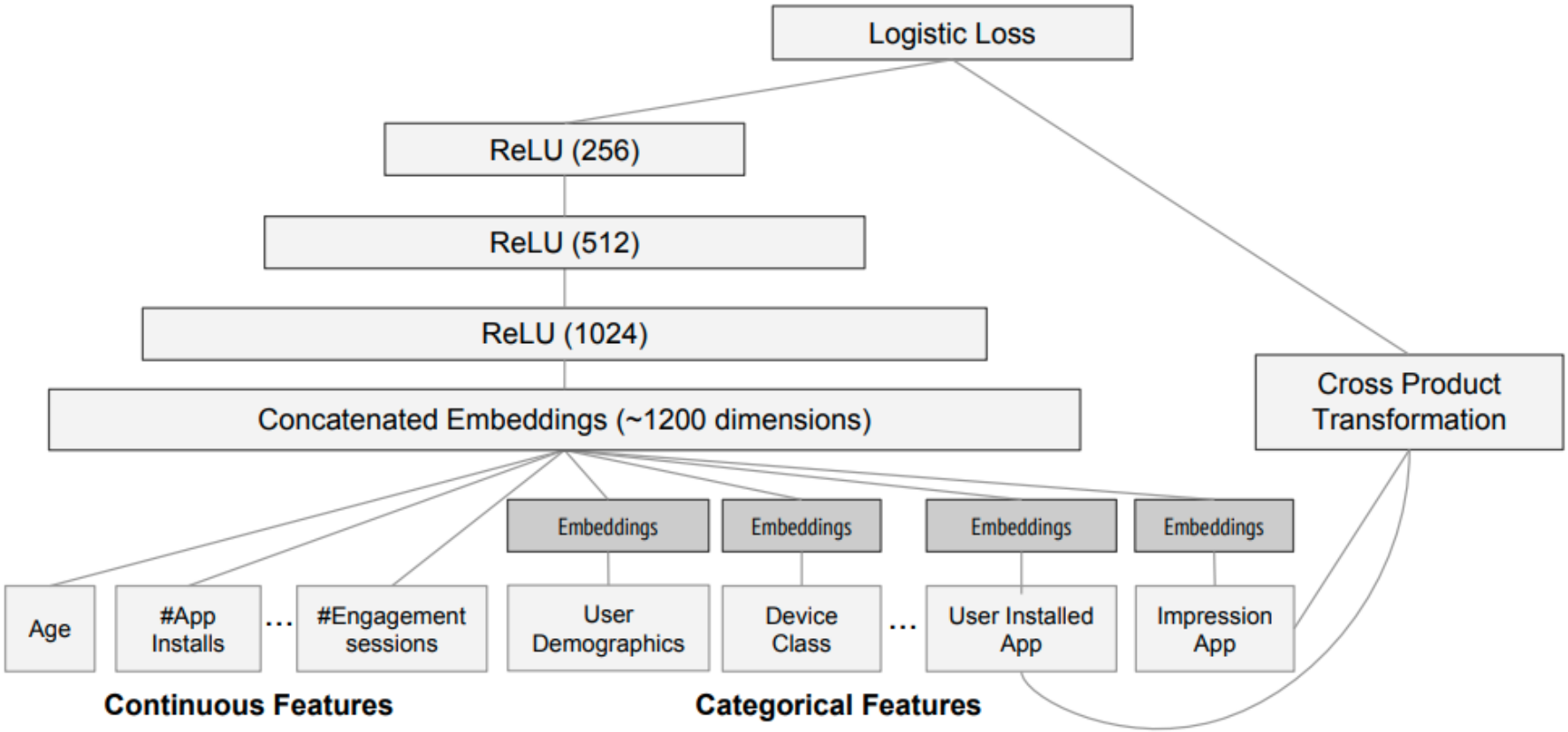
For other features (for example, numerical), embedding layers are not used.

Wide vs Wide & Deep vs Deep



Memorization vs Memorization & Generalization vs Generalization

Network Architecture



Joint training

The wide component and deep component are combined using a weighted sum of their output log odds as the prediction, which is then fed to one common logistic loss function for joint training.

There is a distinction between joint training and ensemble:

- Ensemble
  - individual models, which are trained separately;
  - predictions are combined only at inference time;
  - each individual model size usually needs to be large;
- Joint training:
  - optimizes all parameters simultaneously by taking both the wide and deep part;
  - the wide part only needs to complement the weaknesses of the deep part with a small number of cross-product feature transformations, rather than a full-size wide model;

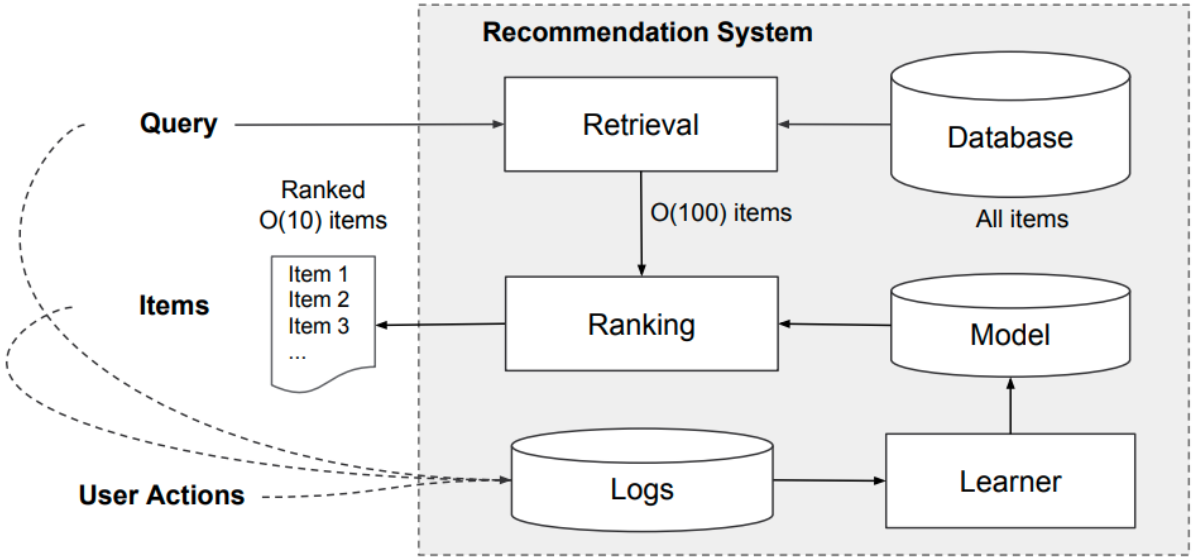
Joint training

Joint training of a Wide & Deep Model is done by backpropagating the gradients from the output to both the wide and deep part of the model simultaneously using mini-batch.

For a logistic regression problem, the model's prediction is:

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

Apps recommendation task

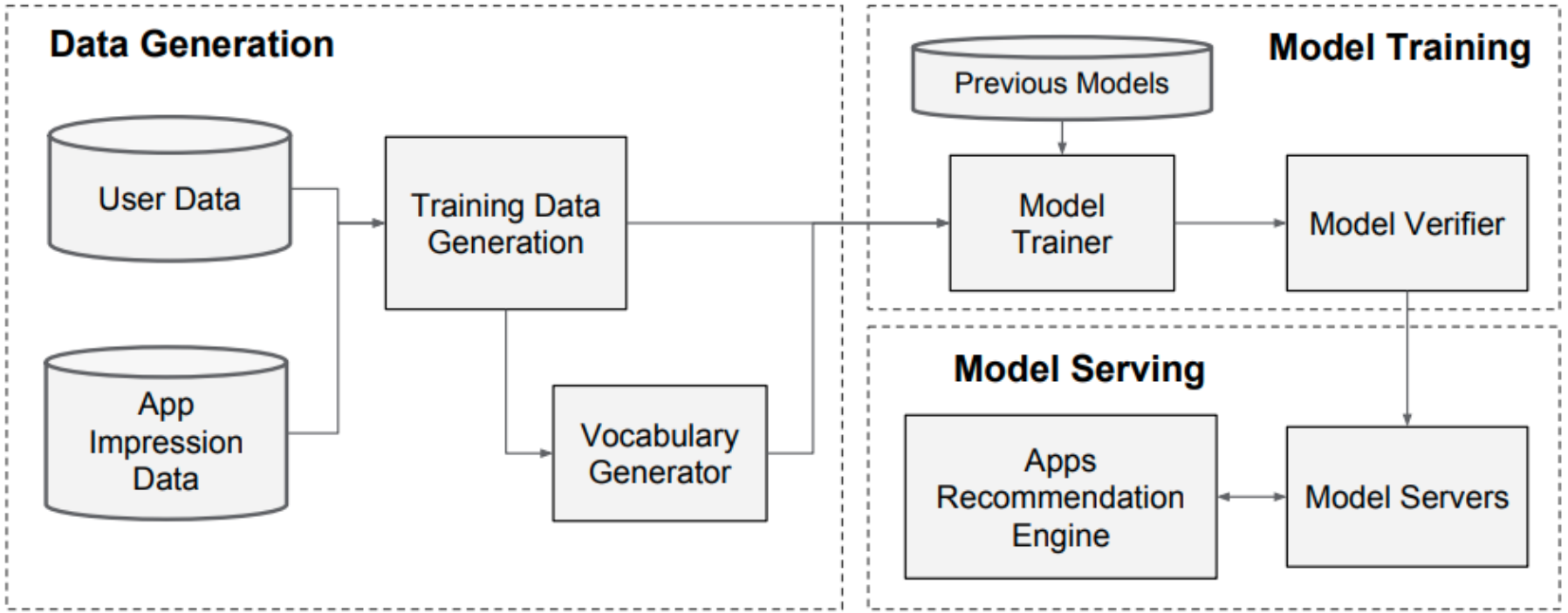


The task is to develop a recommender system productionized on Google Play, a mobile app store with over one billion active users and over one million apps.

Each example corresponds to one impression. The target is app acquisition:

- 1, if the impressed app was installed
- 0, otherwise

Apps recommendation pipeline



Experiment Results

Online Acquisition Gain is relative to the control.

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

Serving latency

Table 2: Serving latency vs. batch size and threads.

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

Conclusion

Memorization and generalization are both important for recommender systems. **Wide linear models** can effectively memorize sparse feature interactions using cross-product feature transformations, while **deep neural networks** can generalize to previously unseen feature interactions through low dimensional embeddings.

The Wide & Deep learning model is able to combine the strengths of both types of model. Online experiment results showed that the Wide & Deep model led to significant improvement on app acquisitions over wide-only and deep-only models.

It should also be noted that this approach can be used not only for recommender systems. Wide & Deep networks are also used in other classification problems, as well as regression, for example, in problems of predicting the expected travel time (ETA).