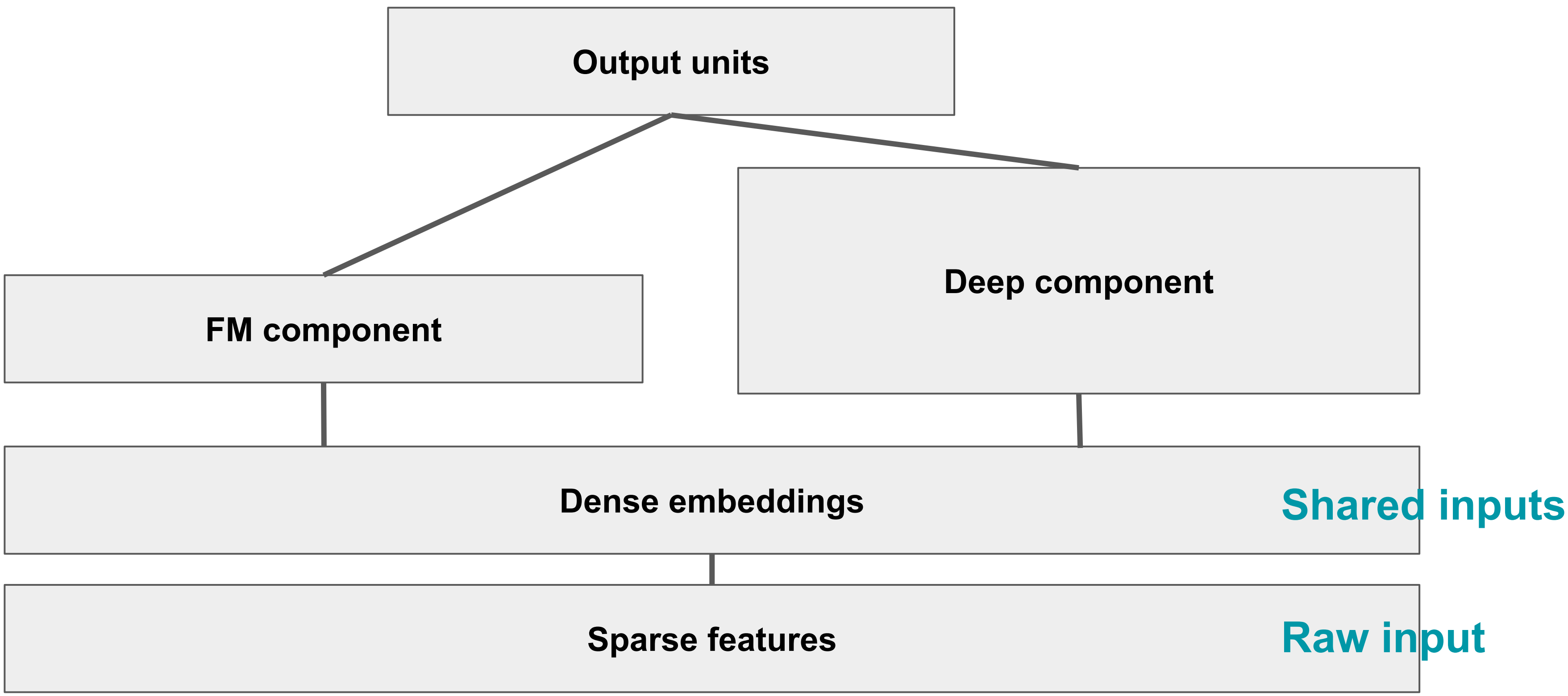
DeepFM: A Factorization-Machine based Neural Network for CTR Prediction

Guo et. al., (IJCAI `17)

Hyunji Choi

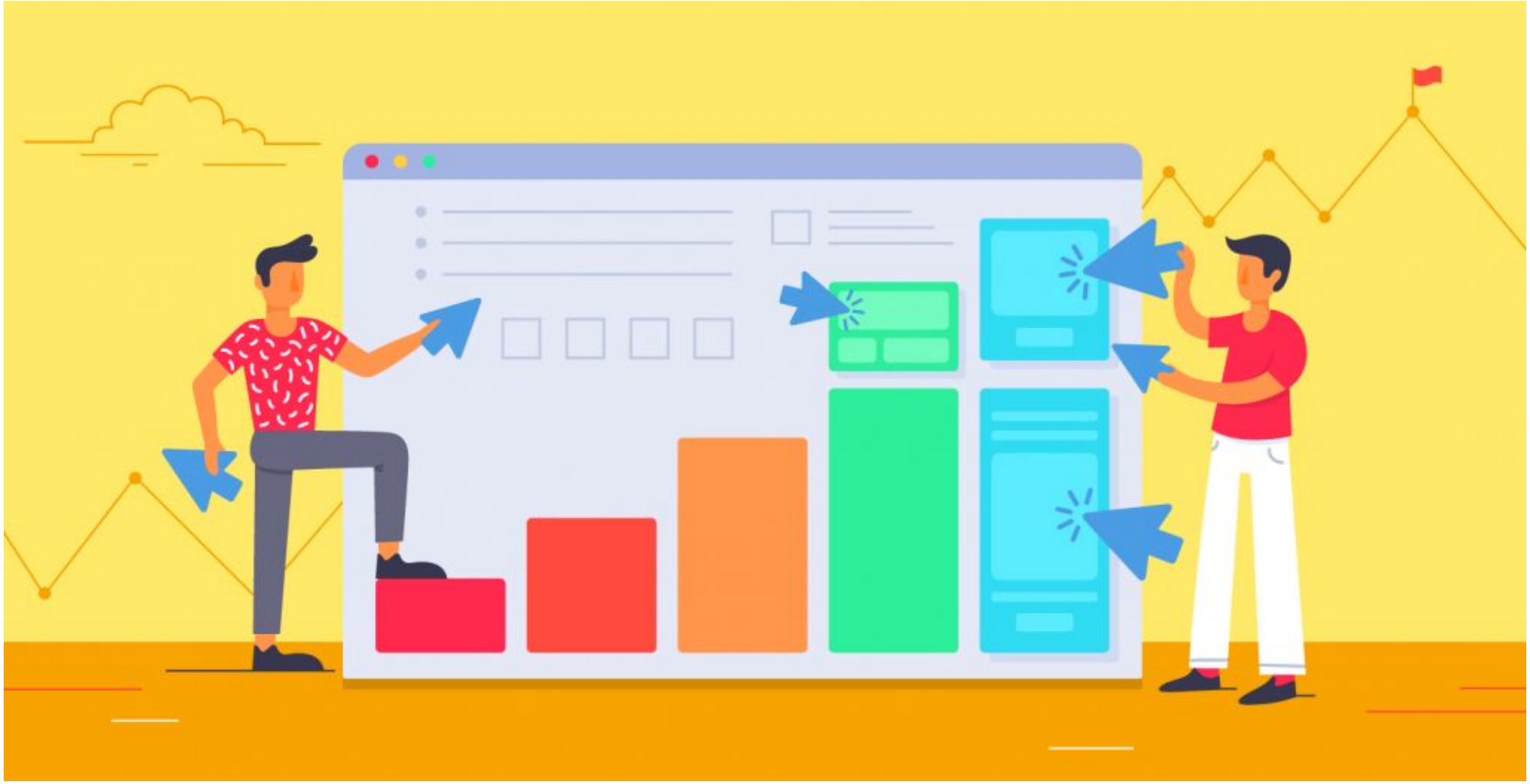
May 17th, 2021

Overview


Output units
Deep component
FM component
Dense embeddings
Shared inputs
Sparse features
Raw input

Click-Through Rate Prediction

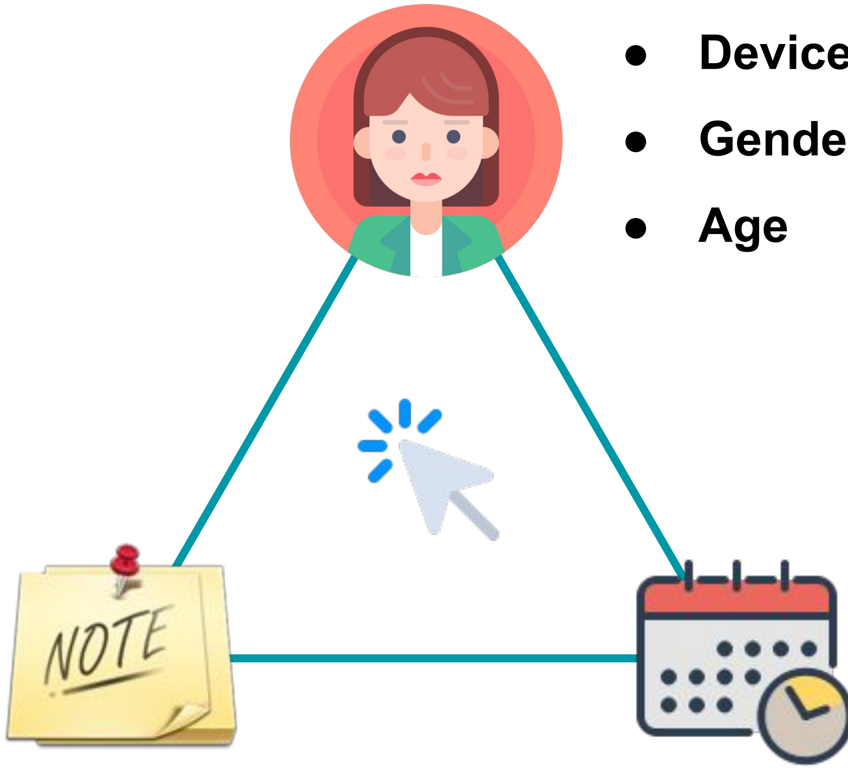
- **Probability a user will click on a recommended item.**
- **Rank the output of recommender by CTR.**



https://blog.creatopy.com/average-display-click-through-rate-ctr/

Implicit feature interactions behind click behaviors

- Device
- Gender
- Age



- UI style
- Cost
- Functions

- Time
- Season
- Location

CTR Prediction Model

- $\hat{y} = CTR\_model(x)$
- **Concatenate categorical & continuous fields.**
- **High-dimensional, sparse feature vector x → dense embeddings.**

User			Gender		Device		Age	App			pdf	time	clicked
1	0	...	1	0	1	0	25	1	0	...	1	13	1
1	0	...	1	0	1	0	27	0	1	...	1	13	0
0	1	...	0	1	1	0	23	0	1	...	1	15	?

Effectively Modeling Feature Interactions

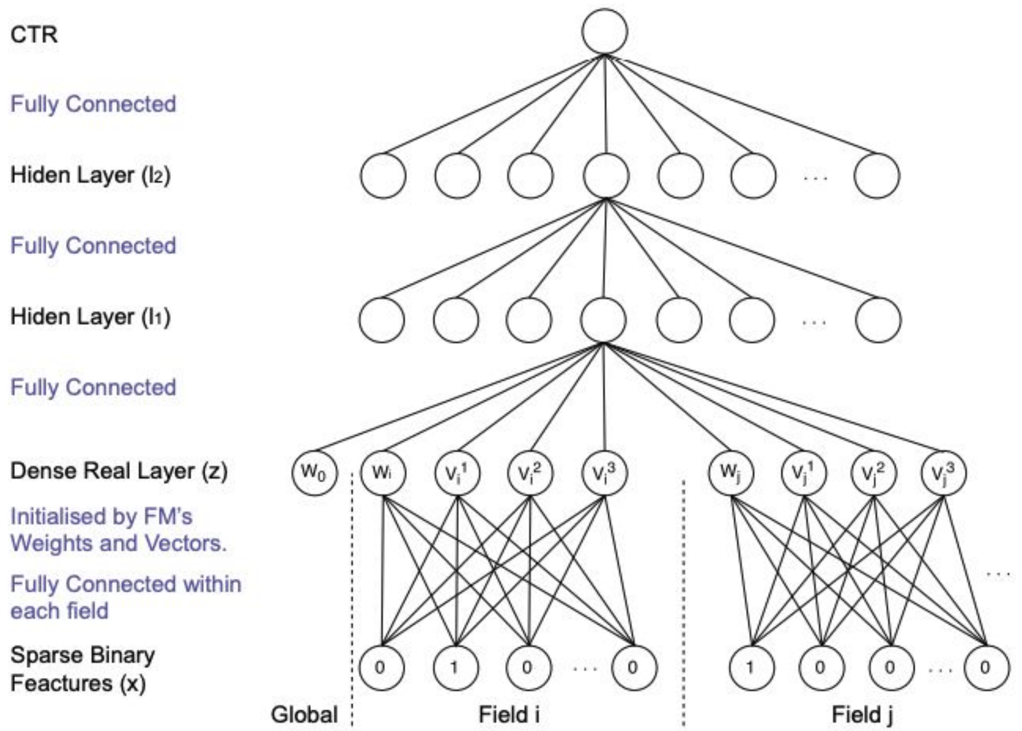
- **Feature: Generalized linear model**
- **Low-order interactions: Factorization Machine**

$$\hat{y}(\mathbf{x}) := w_0 + \sum_{i=1}^{n} w_i \, x_i + \sum_{i=1}^{n} \sum_{j=i+1}^{n} \langle \mathbf{v}_i, \mathbf{v}_j \rangle \, x_i \, x_j$$

- **High-order interactions: DNN-based models**
  - FNN (Zhang et al., 2016)
  - PNN (Qu et al., 2016)
  - Wide & Deep (Cheng et al., 2016)

DNN-based CTR Prediction Models: FNN

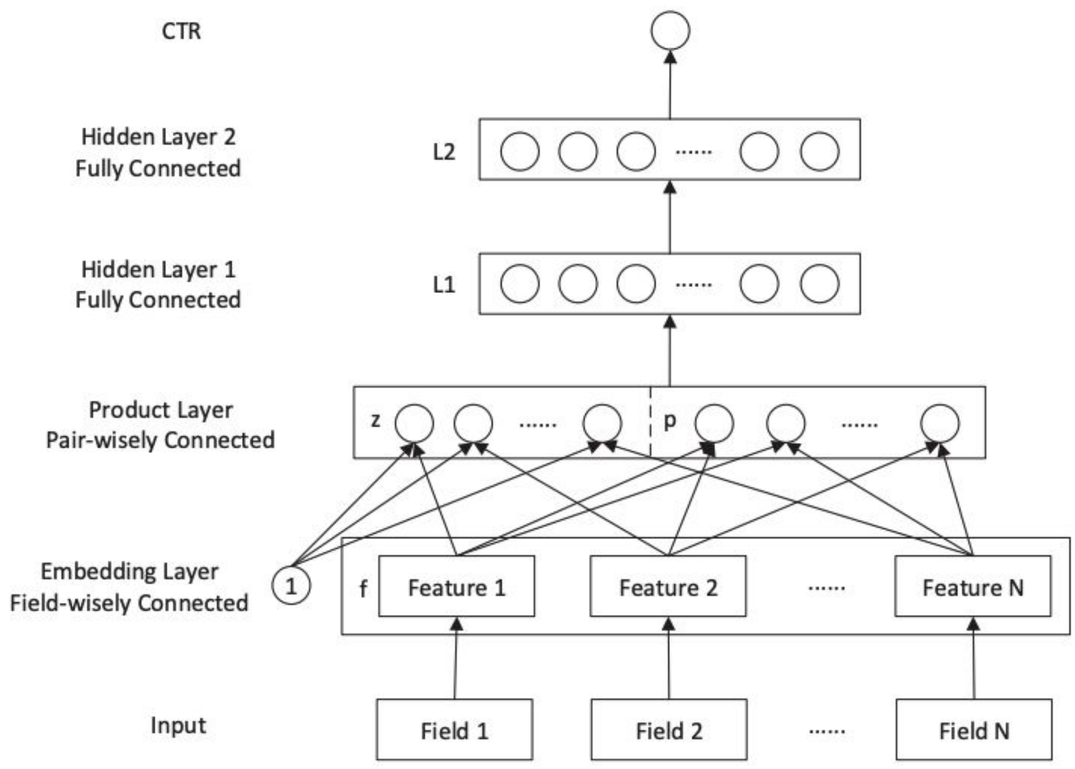
- **FM pre-training → DNN**
- **Factorization-machine supported NN**



  - Training overhead.
  - Embedding parameters over affected by FM.
  - Captures only high-order feature interactions.

DNN-based CTR Prediction Models: PNN

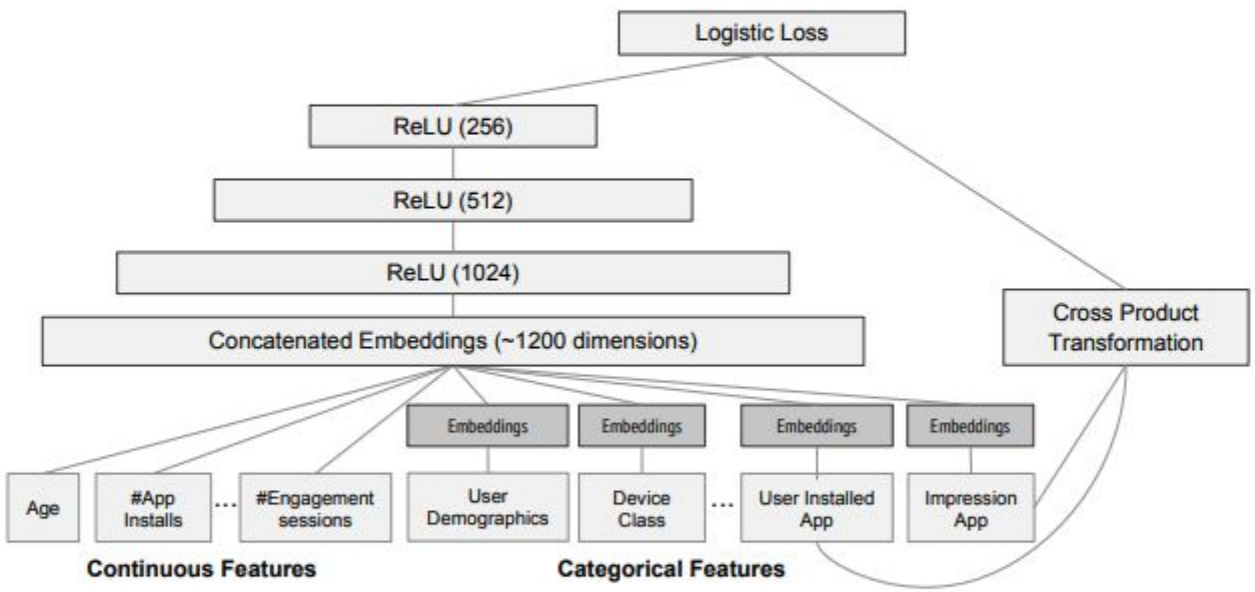
- **Product layer → DNN**
- **Product-based NN**



  - Training complexity.
  - Captures only high-order feature interactions.

DNN-based CTR Prediction Models: Wide & Deep

- **FM + DNN simultaneously.**
- **Captures both low- and high-order interactions.**



  - Needs expertise feature engineering.
  - Separate types of inputs.

Both low- and high-order feature interactions are important

- **Wide & Deep outperformed Wide and Deep.**

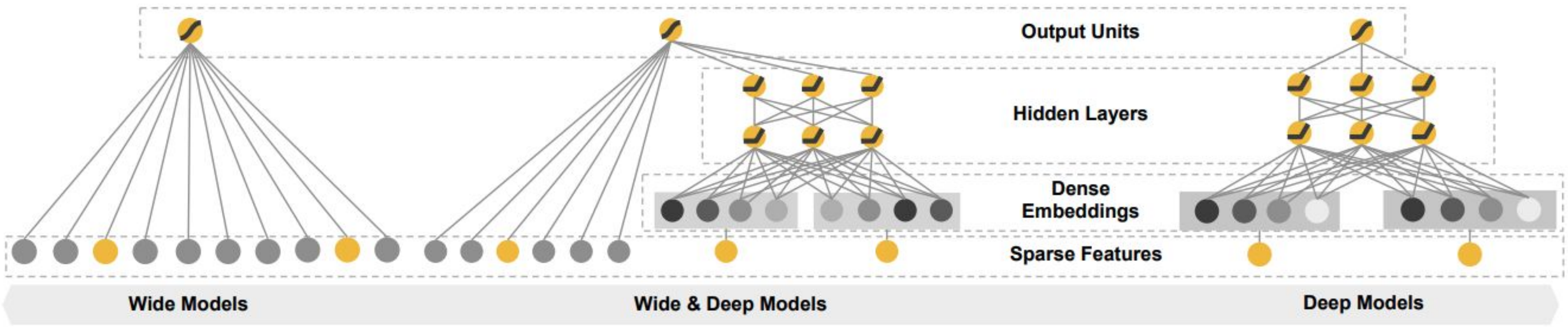
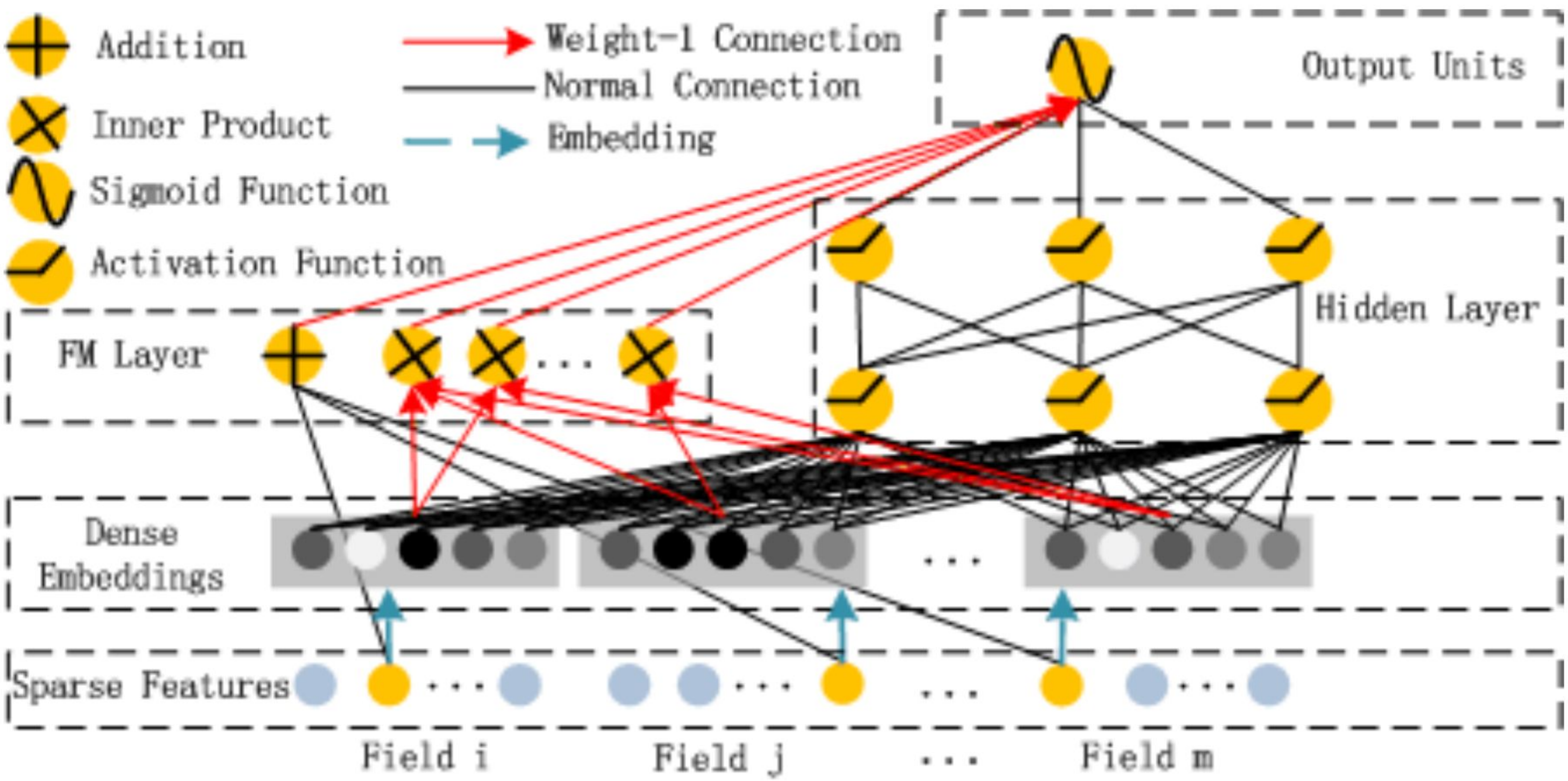


Table 1: Offline & online metrics of different models. Online Acquisition Gain is relative to the control.

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

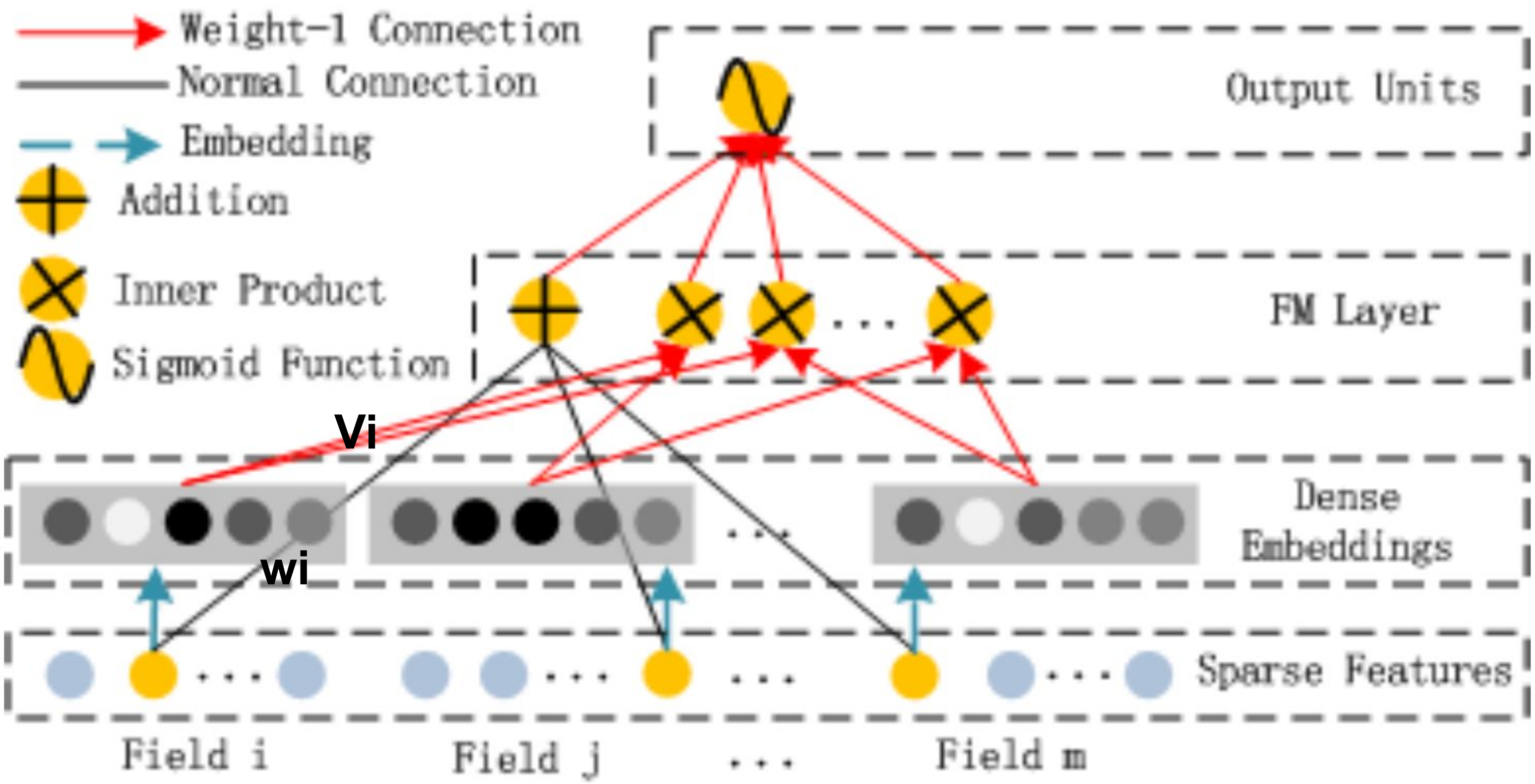
Approach

- **FM and deep component share the same input.**
- $\hat{y} = sigmoid(y_{FM} + y_{DNN})$



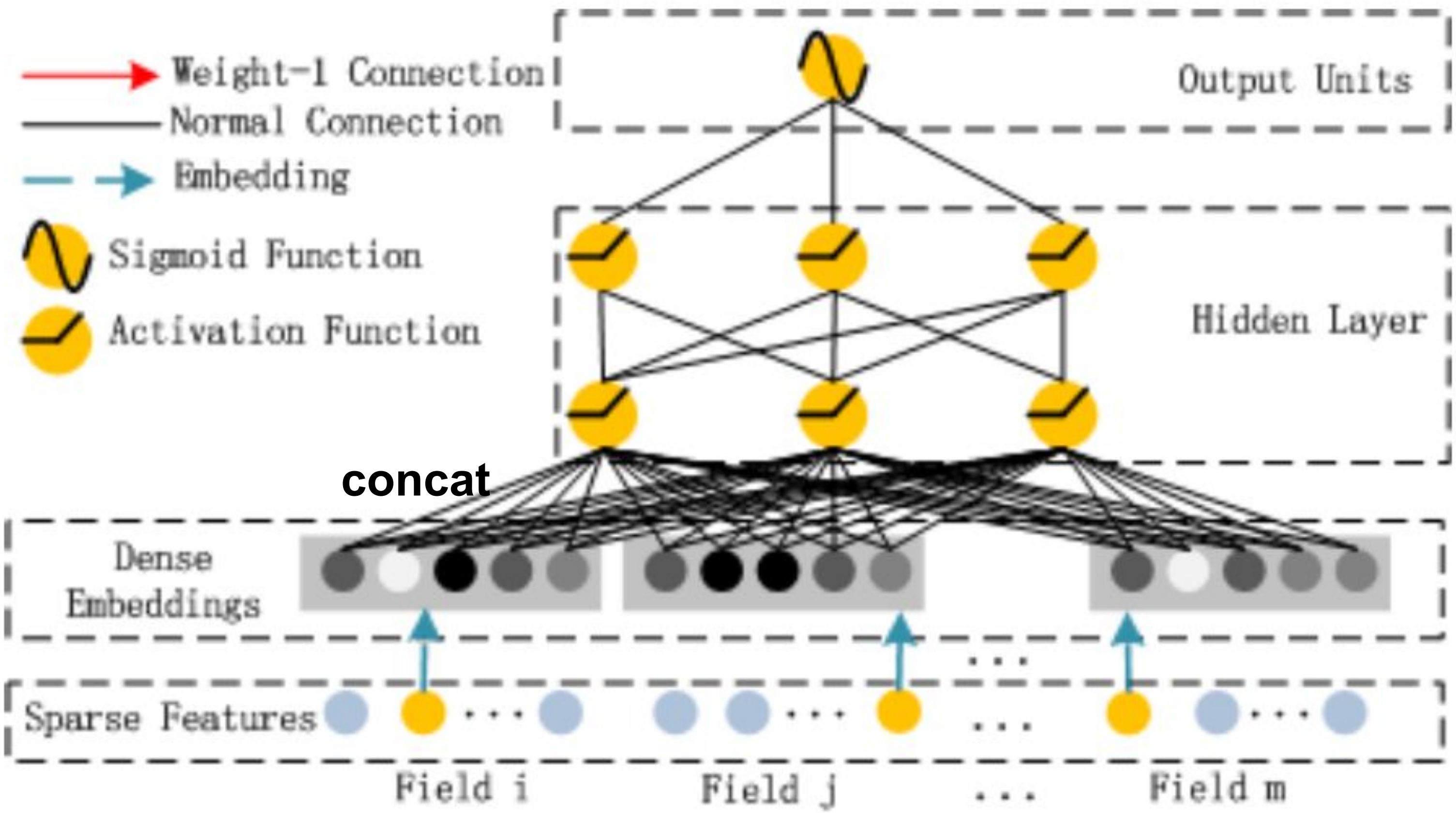
FM Component

- $y_{FM} = \langle w, x \rangle + \sum_{j_1=1}^{d} \sum_{j_2=j_1+1}^{d} \langle V_i, V_j \rangle x_{j_1} \cdot x_{j_2}$



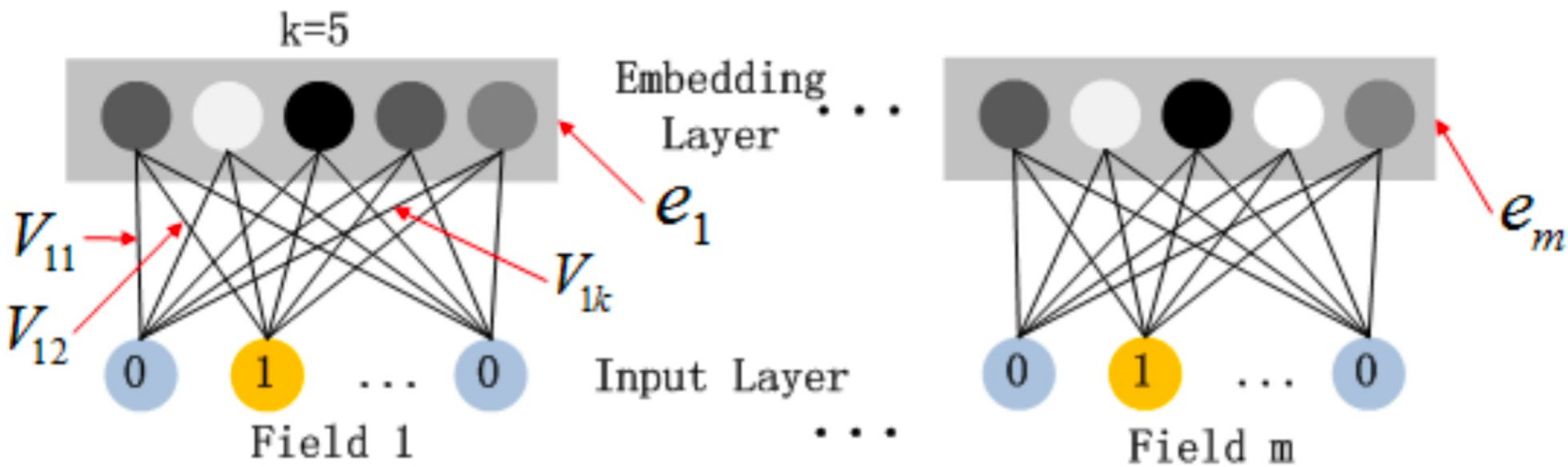
Deep Component

- $y_{DNN} = \sigma(W^{|H|+1} \cdot a^H + b^{|H|+1})$



Deep Component

- **Structure of Embedding Layer**
- **FM per field: output ei length is the same (k).**



Data and Metric

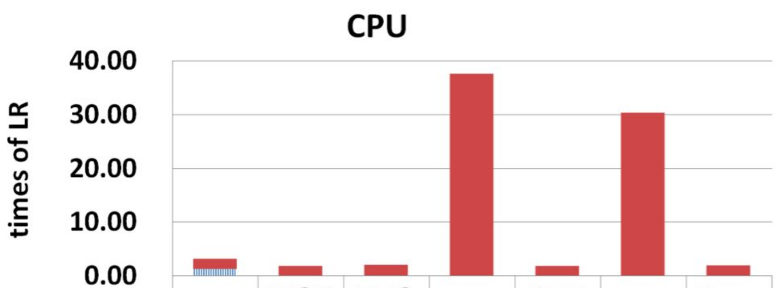
- **Criteo: 45 million users' click records.**
- **Company***
    - 7 consecutive days of users' click records from the App Store for training.
    - Next 1 day for testing.
    - 1 billion records with app, user, and context features.

- **AUC**
- **Logloss**

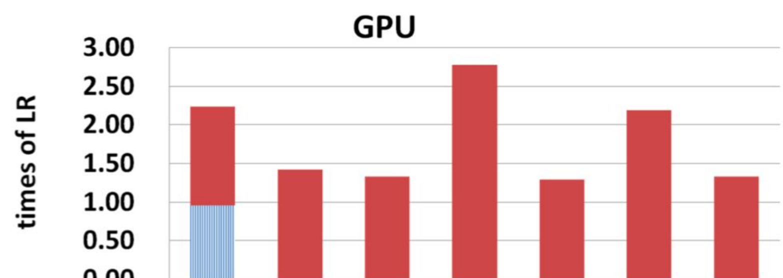
Model Comparison

- **LR**
- **FM**
- **FNN**
- **PNN (inner / outer / both)**
- **Wide & Deep**
- **Wide & Deep replaced LR with FM**
- **DeepFM**

DeepFM Training is Efficient



	FNN	LR&DNN	FM&DNN	IPNN	OPNN	PNN*	DeepFM
training	1.89	1.89	2.15	37.61	1.92	30.44	2.00
pre-training	1.32	0.00	0.00	0.00	0.00	0.00	0.00



	FNN	LR&DNN	FM&DNN	IPNN	OPNN	PNN*	DeepFM
training	1.28	1.41	1.32	2.77	1.29	2.19	1.32
pre-training	0.95	0.00	0.00	0.00	0.00	0.00	0.00

- **Pre-training is inefficient.**
- **DeepFM is efficient on both CPU and GPU.**

DeepFM outperforms other models in CTR prediction

Table 2: Performance on CTR prediction.

	Company∗		Criteo	
	AUC	LogLoss	AUC	LogLoss
LR	0.8640	0.02648	0.7686	0.47762
FM	0.8678	0.02633	0.7892	0.46077
FNN	0.8683	0.02629	0.7963	0.45738
IPNN	0.8664	0.02637	0.7972	0.45323
OPNN	0.8658	0.02641	0.7982	0.45256
PNN∗	0.8672	0.02636	0.7987	0.45214
LR & DNN	0.8673	0.02634	0.7981	0.46772
FM & DNN	0.8661	0.02640	0.7850	0.45382
DeepFM	**0.8715**	**0.02618**	**0.8007**	**0.45083**

- **LR vs others: learning feature interactions is important.**
- **FM / FNN / PNN vs DeepFM: learning both low- and high-order interactions is critical.**
- **LR&DNN vs DeepFM: sharing feature embedding improves the performance.**

Hyper-Parameter Study (Network)

- **Over-complicated model is easy to overfit.**
    - Number of neurons per layer: 200 or 400.
    - Number of hidden layers: about 3.
- **Constant network shape is empirically better.**

Conclusion

- **DeepFM outperforms the state-of-the-art models.**
  - Learns both high- and low-order feature interactions.
  - Shares feature embedding to avoid feature engineering.
- **DeepFM shows comparable efficiency.**
  - No pre-training.
  - Moderate training complexity.