Automated Machine Learning for Recommendations: Fundamentals and Advances

Xiangyu Zhao
City University of Hong Kong
xianzhao@cityu.edu.hk

Wenqi Fan
The Hong Kong Polytechnic University
wenqifan03@gmail.com

Huifeng Guo
Huawei Noah's Ark Lab
huifeng.guo@huawei.com

Bo Chen
Huawei Noah's Ark Lab
chenbo116@huawei.com

Yejing Wang
City University of Hong Kong
adave631@gmail.com

Ruiming Tang
Huawei Noah's Ark Lab
tangruiming@huawei.com

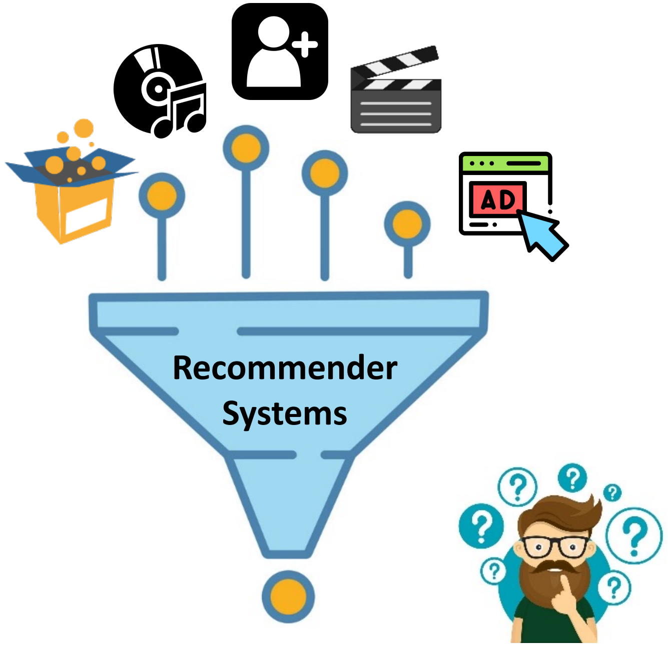
Tutorial Website (Slides): https://advanced-recommender-systems.github.io/AutoML-Recommendations/

Automated Machine Learning for Deep Recommender Systems: A Survey, arXiv:2204.01390

Recommender Systems

Age of Information Explosion

Information overload



Recommend item X to user

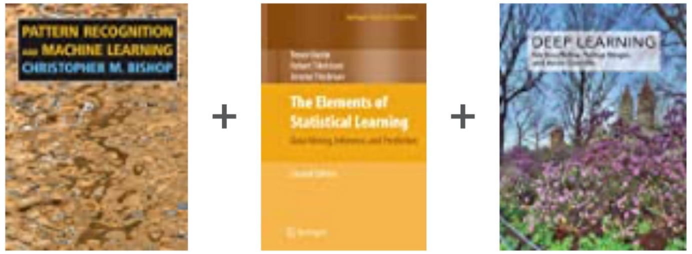
Items can be Products, News, Movies, Videos, Friends, etc.

Recommender Systems

Recommendation has been widely applied in online services:
- **E-commerce**, Content Sharing, Social Networking ...

Product Recommendation

Frequently bought together



A + B + C

Total price: $208.9

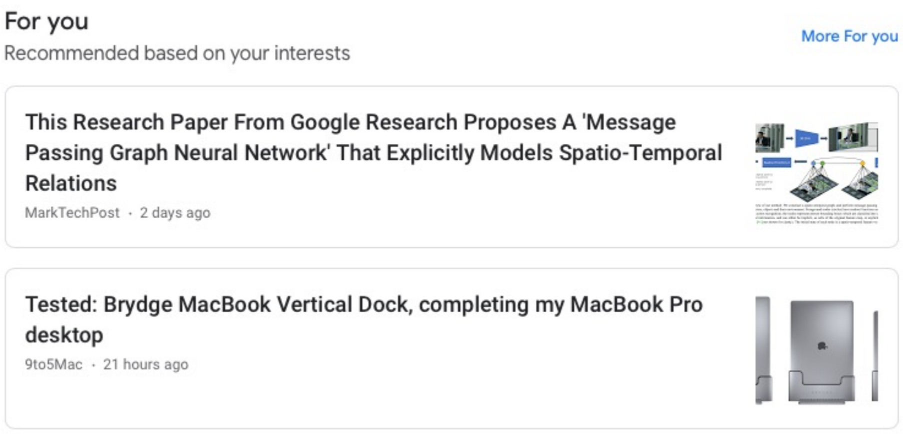
Add all three to Cart

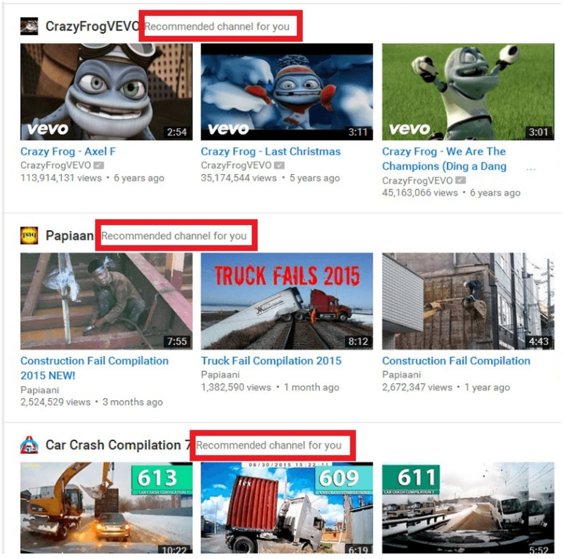
Add all three to List

Recommender Systems

Recommendation has been widely applied in online services:
- E-commerce, **Content Sharing**, Social Networking ...

News/Video/Image Recommendation

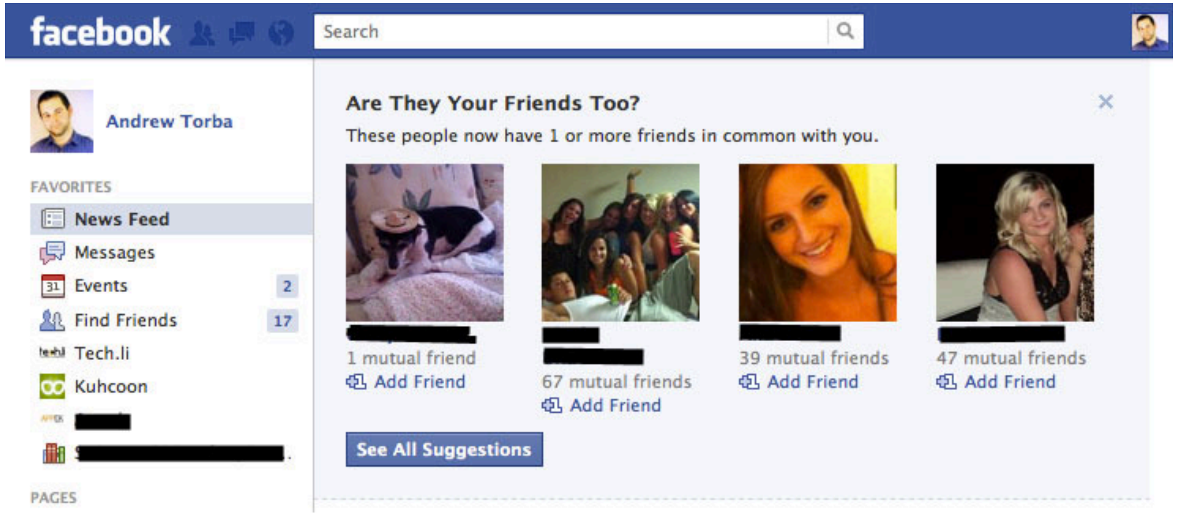




Recommender Systems

Recommendation has been widely applied in online services:
- E-commerce, Content Sharing, **Social Networking** ...

Friend Recommendation



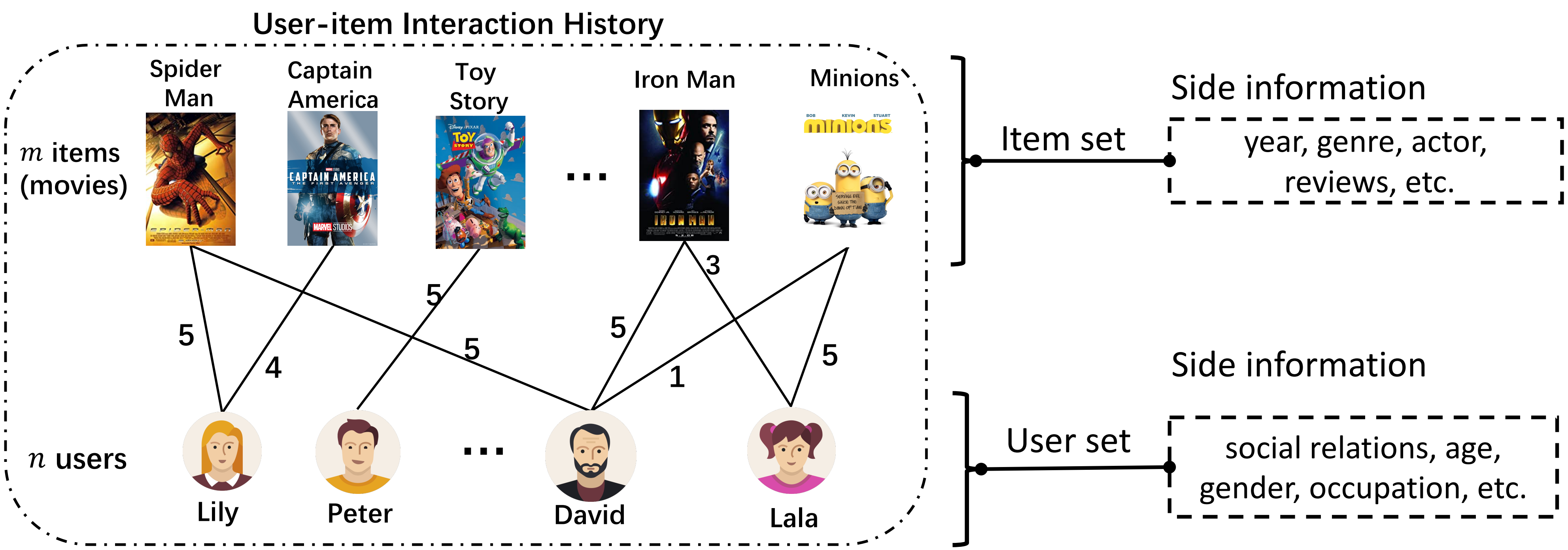
Problem Formulation

INPUT

Historical user-item interactions or additional side information (e.g., social relations, item's knowledge, etc.)

OUTPUT

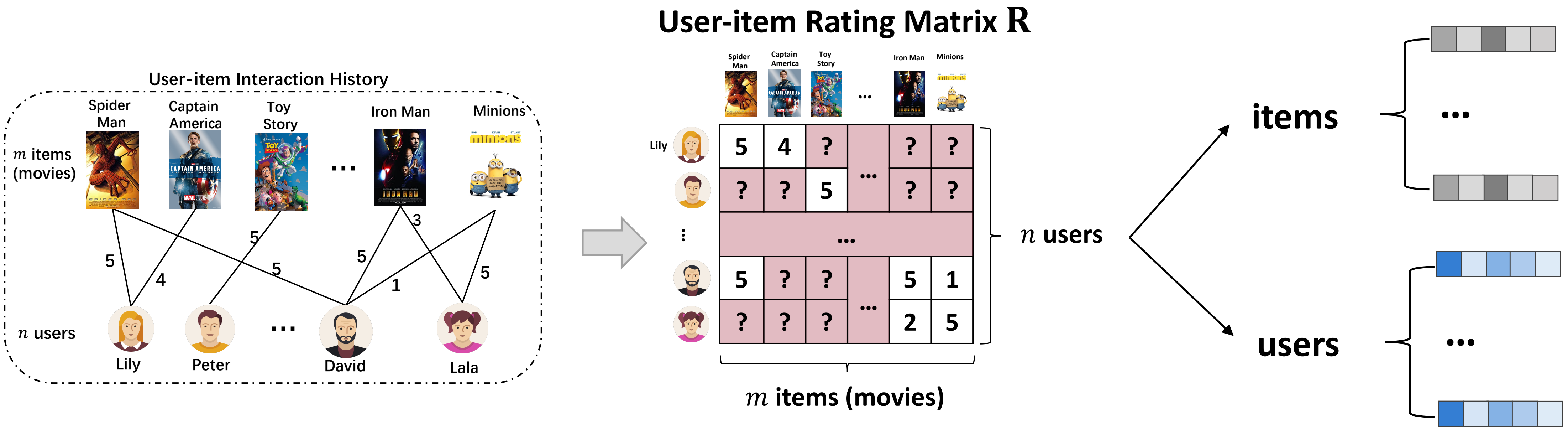
Predict how likely a user would interact with a target Item (e.g., click, view, or purchase)



Recommender Systems

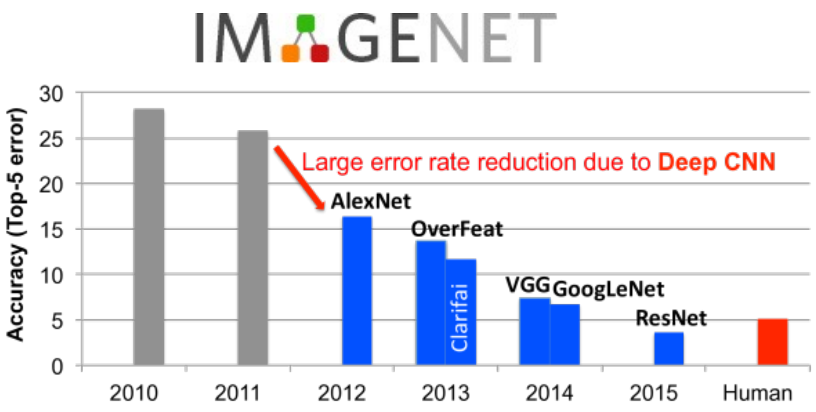
- Collaborative Filtering (CF) is the most well-known technique for recommendation.
  - Similar users (with respect to their historical interactions) have similar preferences.
  - Modelling users' preference on items based on their past interactions (e.g., ratings and clicks).

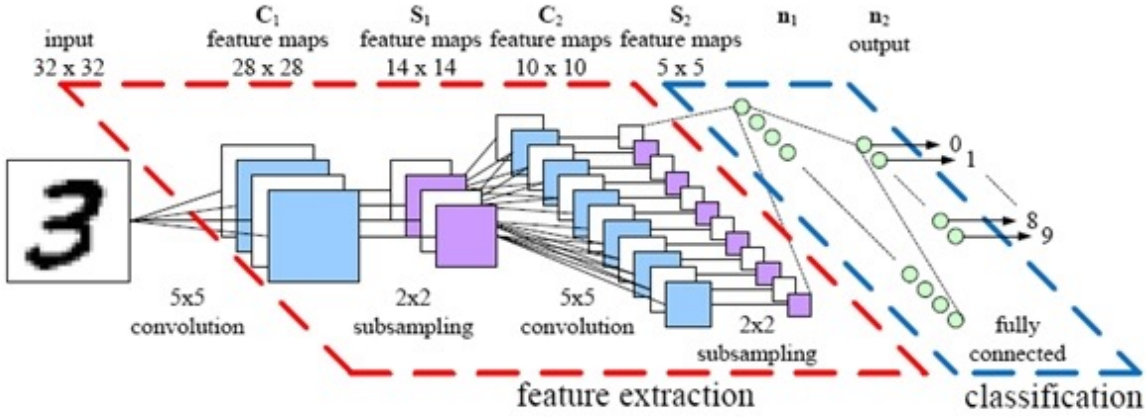
- Learning representations of users and items is the key of CF.

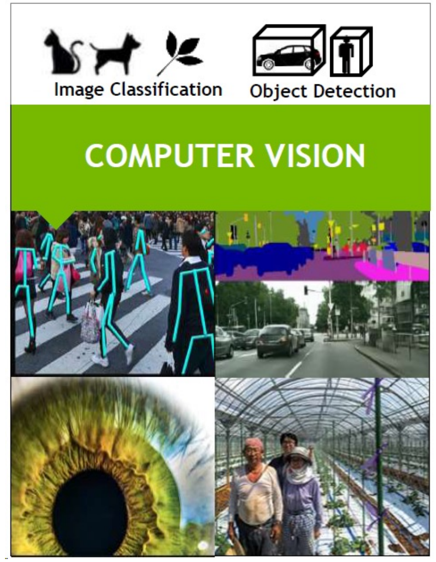


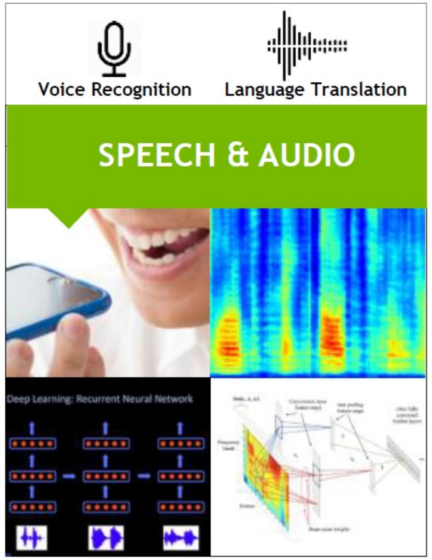
Task: predicting missing movie ratings in Netflix.

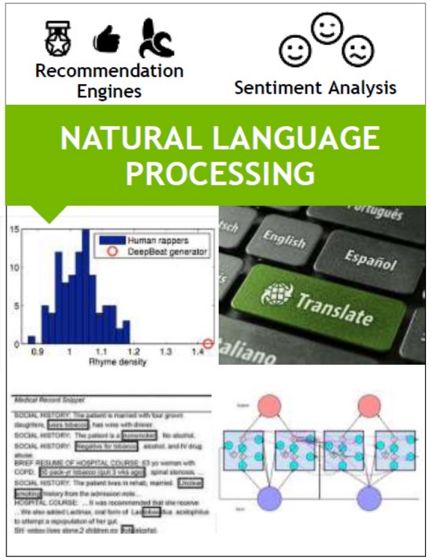
Deep Learning is Changing Our Lives



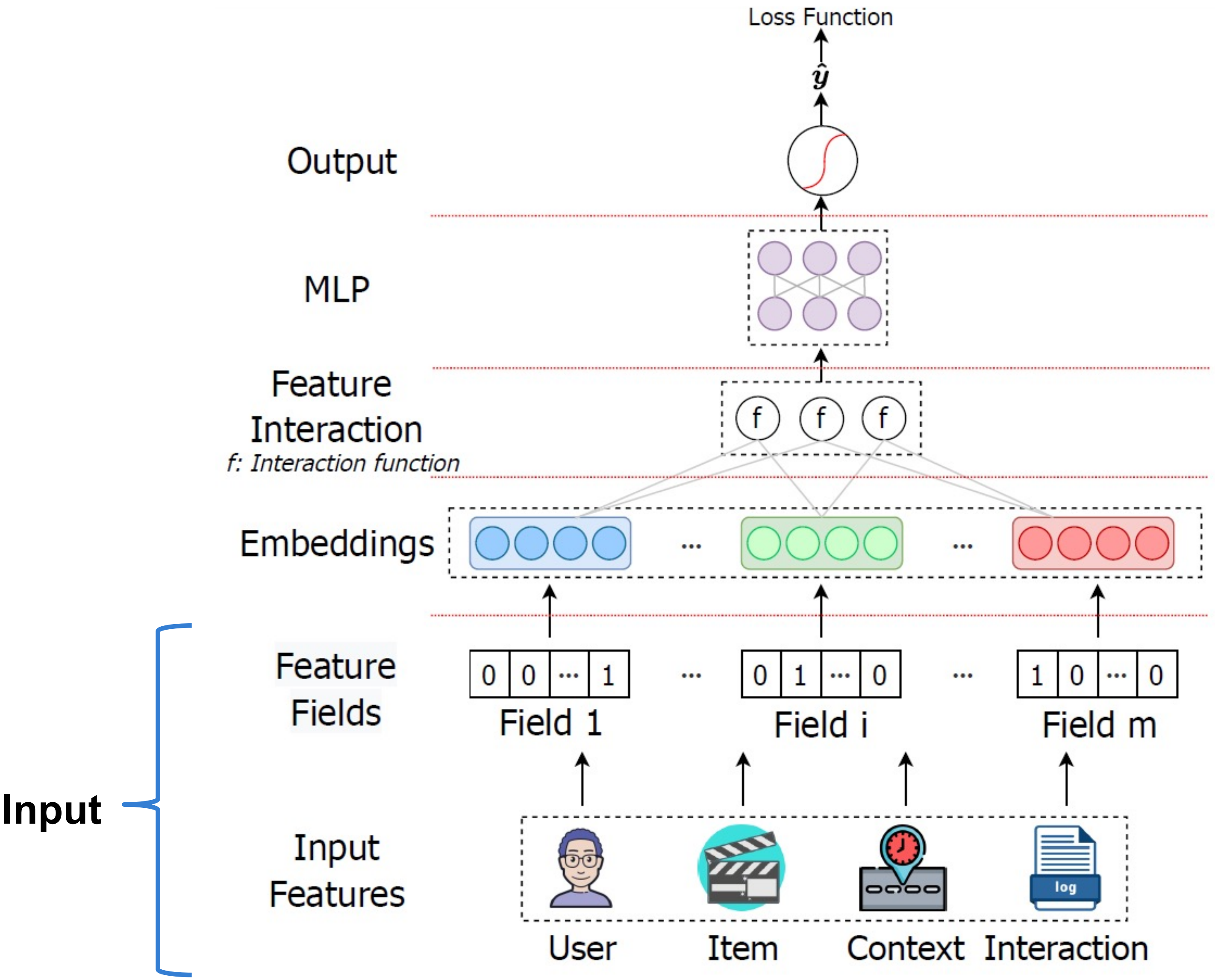




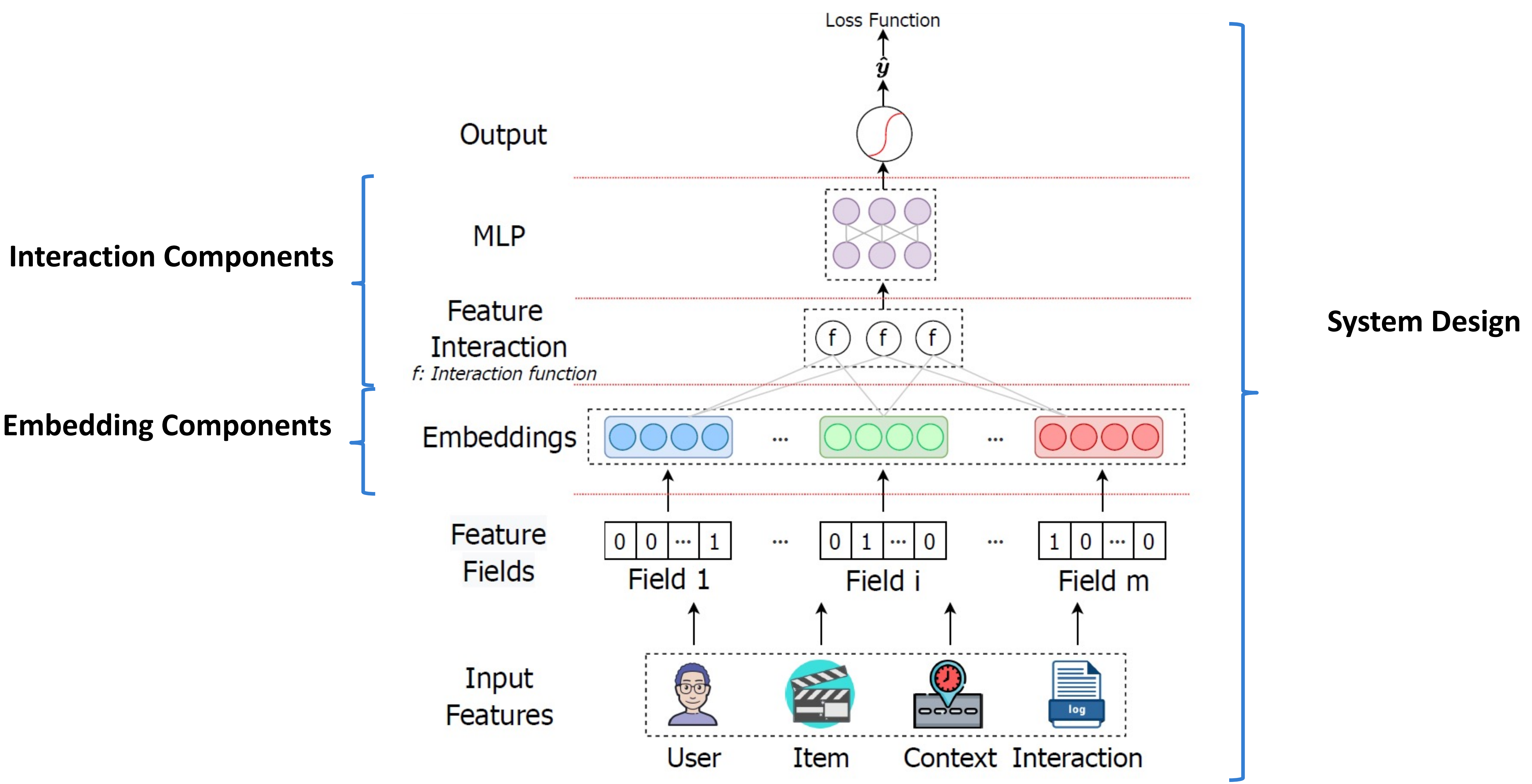




Deep Recommender Architecture



Deep Recommender Architecture



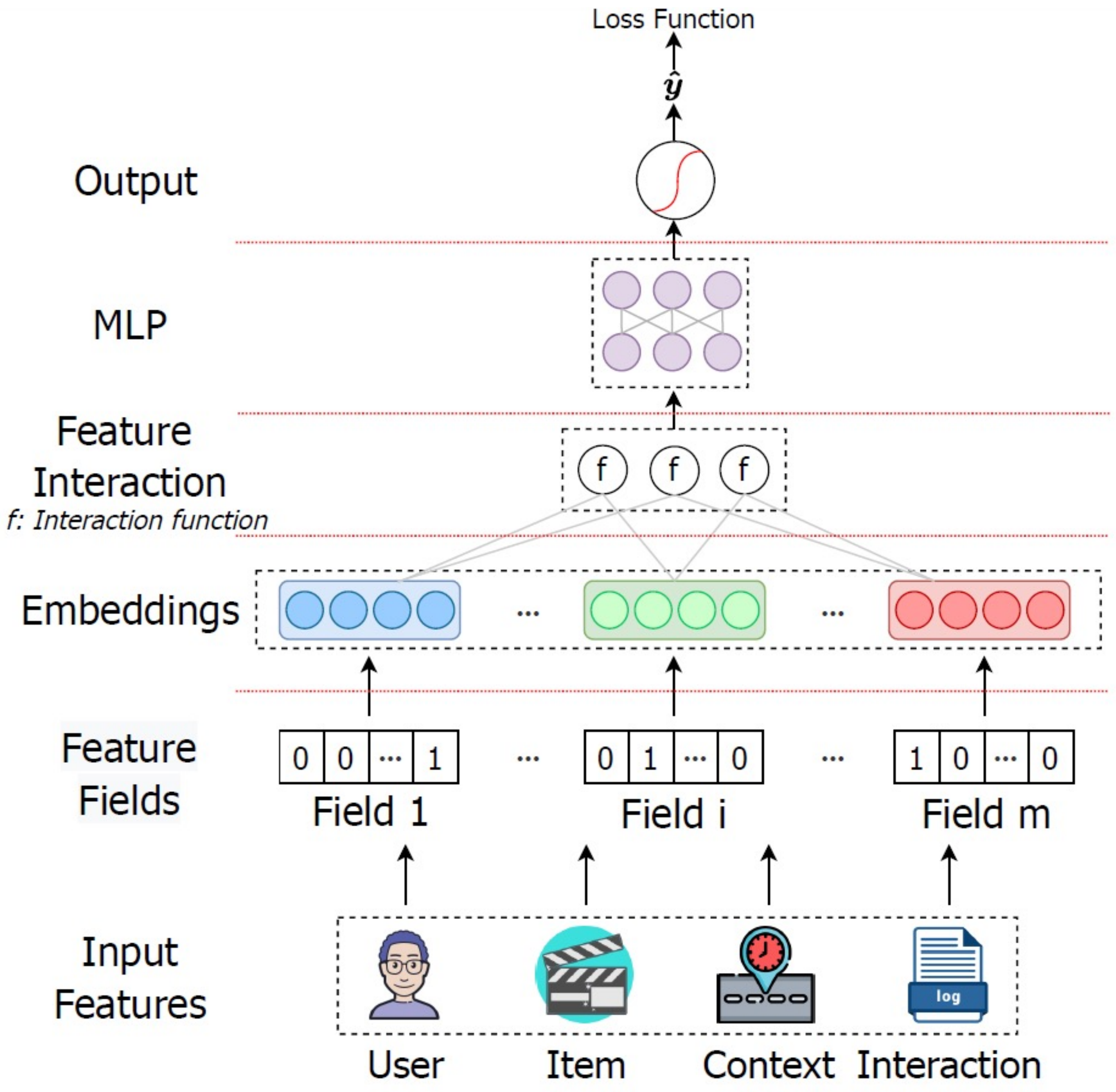
Deep Recommender Architecture

Interaction Components

Pooling, convolution, and the number of layers, inner product, outer product, convolution, etc.

Embedding Components

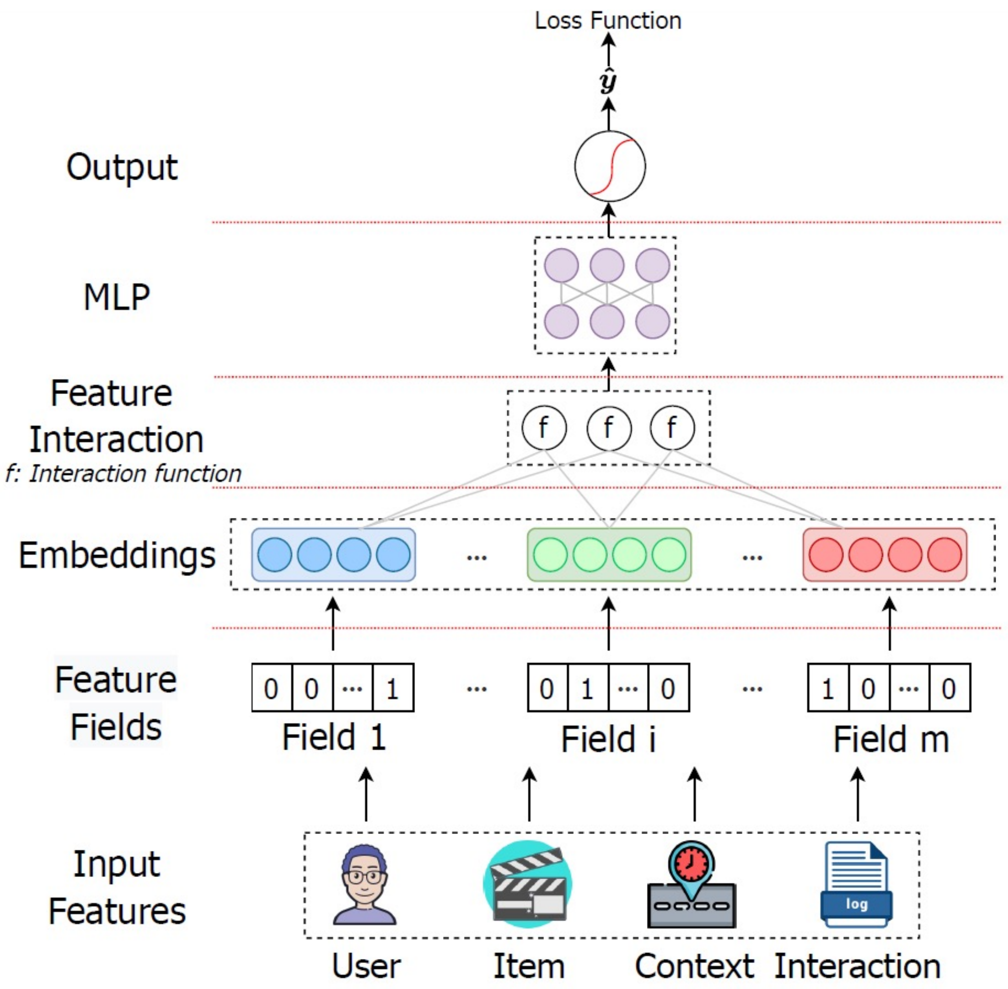
High/low-frequency features embedding sizes



System Design

hardware infrastructure, data pipeline, information transfer, implementation, deployment, optimization, evaluation, etc.

Deep Recommender Architecture



✓ **Advantages**

- Feature representations of users and items
- Non-linear relationships between users and items

➢ **Manually designed architecture:**

- extensive expertise
- substantial engineering labor and time cost
- human bias and error

AutoML for Deep Recommender Systems (DRS)

- Deep architectures are designed by the machine automatically
- Advantages
  - ✓ Less expert knowledge
  - ✓ Saving time and efforts
  - ✓ Different data ->different architectures

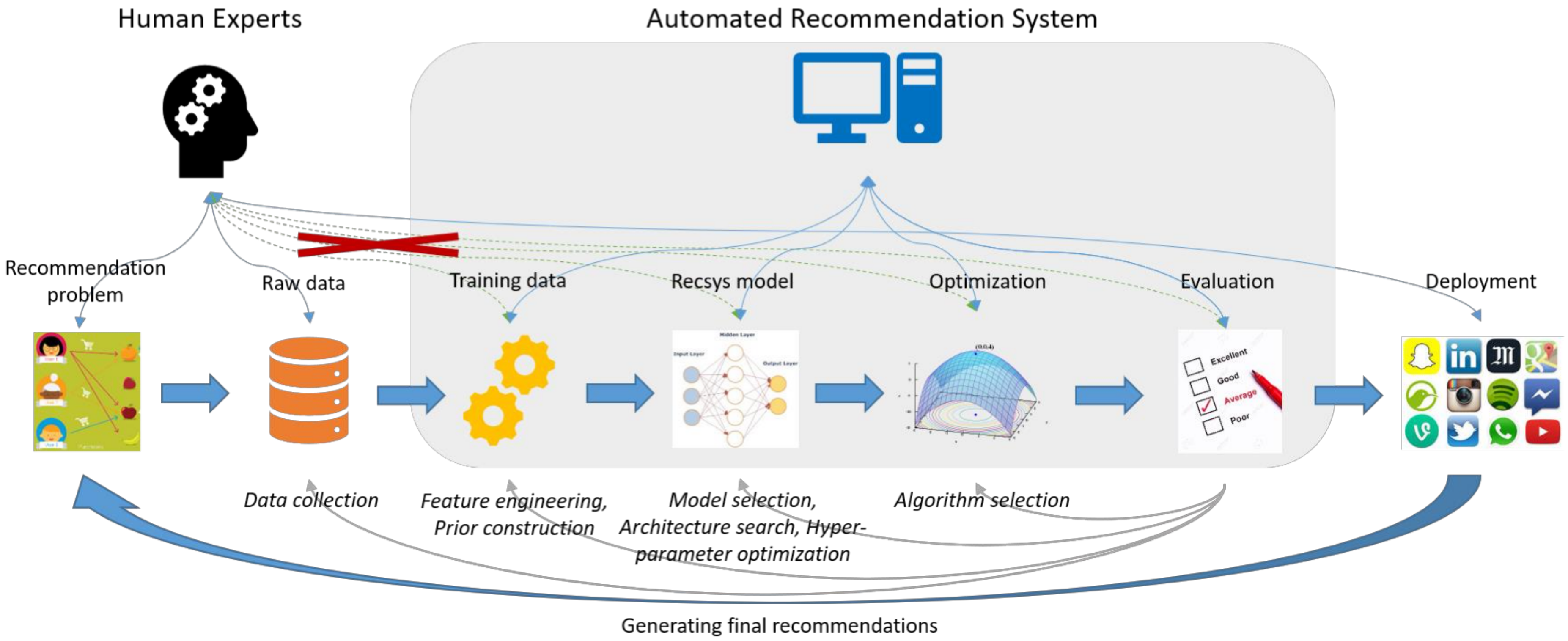


Image: Huan Zhao

- **Agenda**
  - ➢Introduction to Deep Recommender System (Wenqi Fan)
  - ➢Preliminary of AutoML (Xiangyu Zhao)
  - ➢DRS Embedding Components (Bo Chen)
  - ➢DRS Interaction Components (Bo Chen)
  - ➢DRS Comprehensive Search & System (Yejing Wang)
  - ➢Conclusion & Future Direction (Xiangyu Zhao)
  - ➢Q&A

Automated Machine Learning for Deep Recommender Systems: A Survey. arXiv:2204.01390

Tutorial Website (Slides): https://advanced-recommender-systems.github.io/AutoML-Recommendations/

Table of Contents

- Introduction
  - Background
- Preliminary of Automated Machine Learning (AutoML)
  - Neural Architecture Search (NAS)
- DRS Embedding Components
- DRS Interaction Components
- DRS Comprehensive Search & System
- Conclusion & Future Direction

Success of Machine Learning

Astronomy

Robotic

Teaching

Material Design

Creative Arts

Energy

Game Play

Search

Chemistry

Health Care

Physics

Image Recognition

Weather Prediction

Product Recommendation

Drug Discovery

Service

Financial Services

Manufacturing

Traffic Prediction

Retail

Credit Assignment

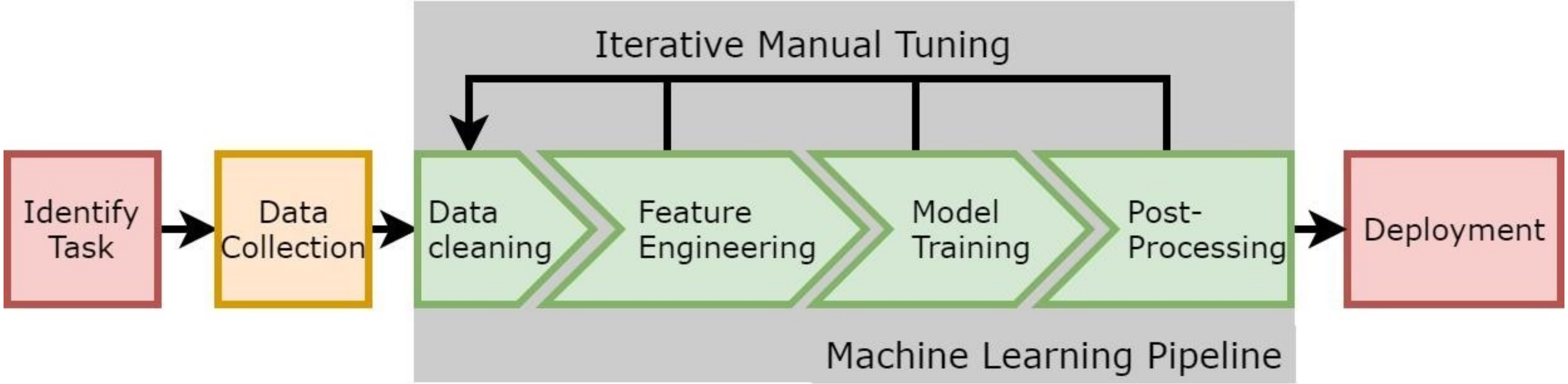
Media

Maintenance Prediction

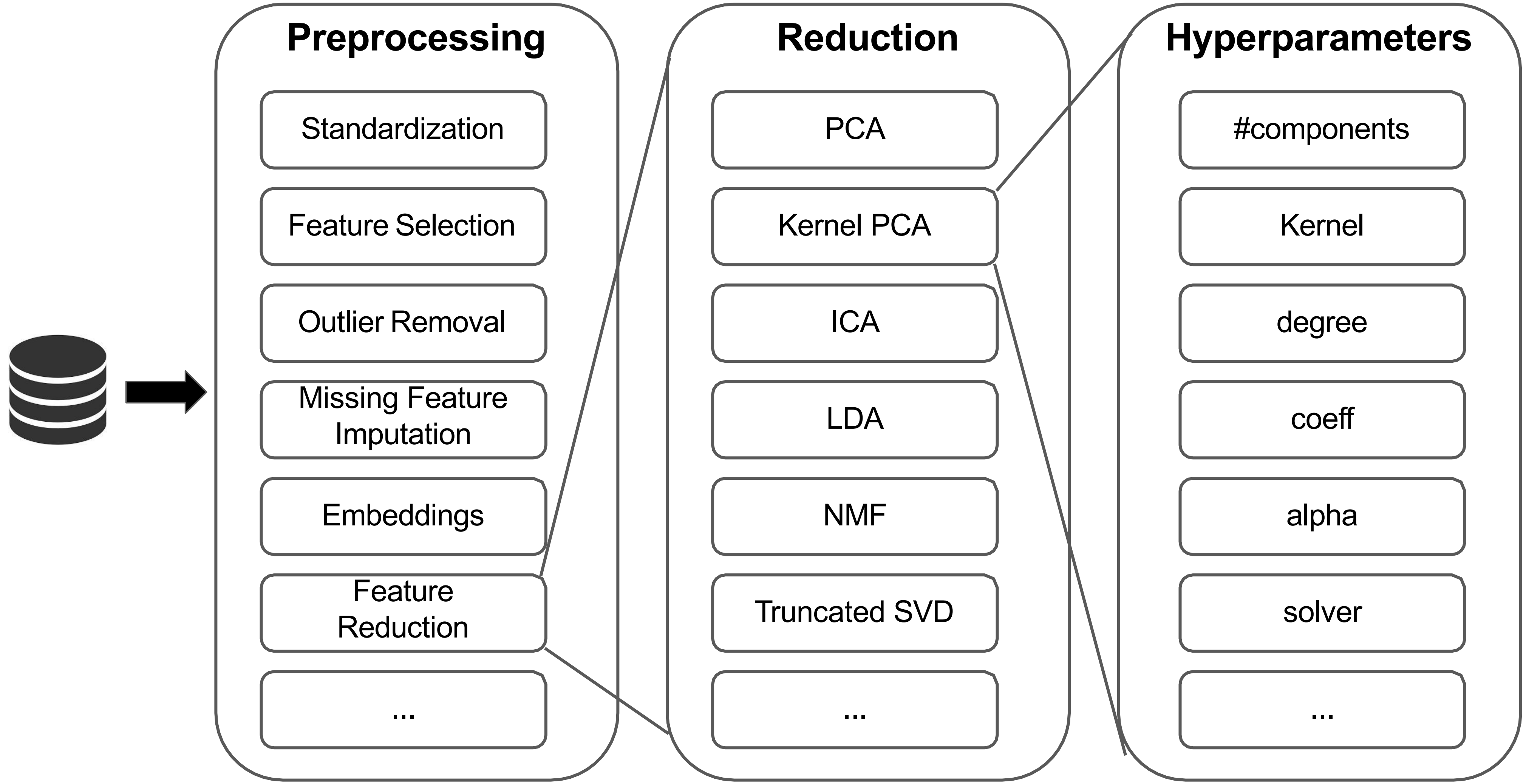
Social Media

Summary Generation

Machine Learning Pipeline

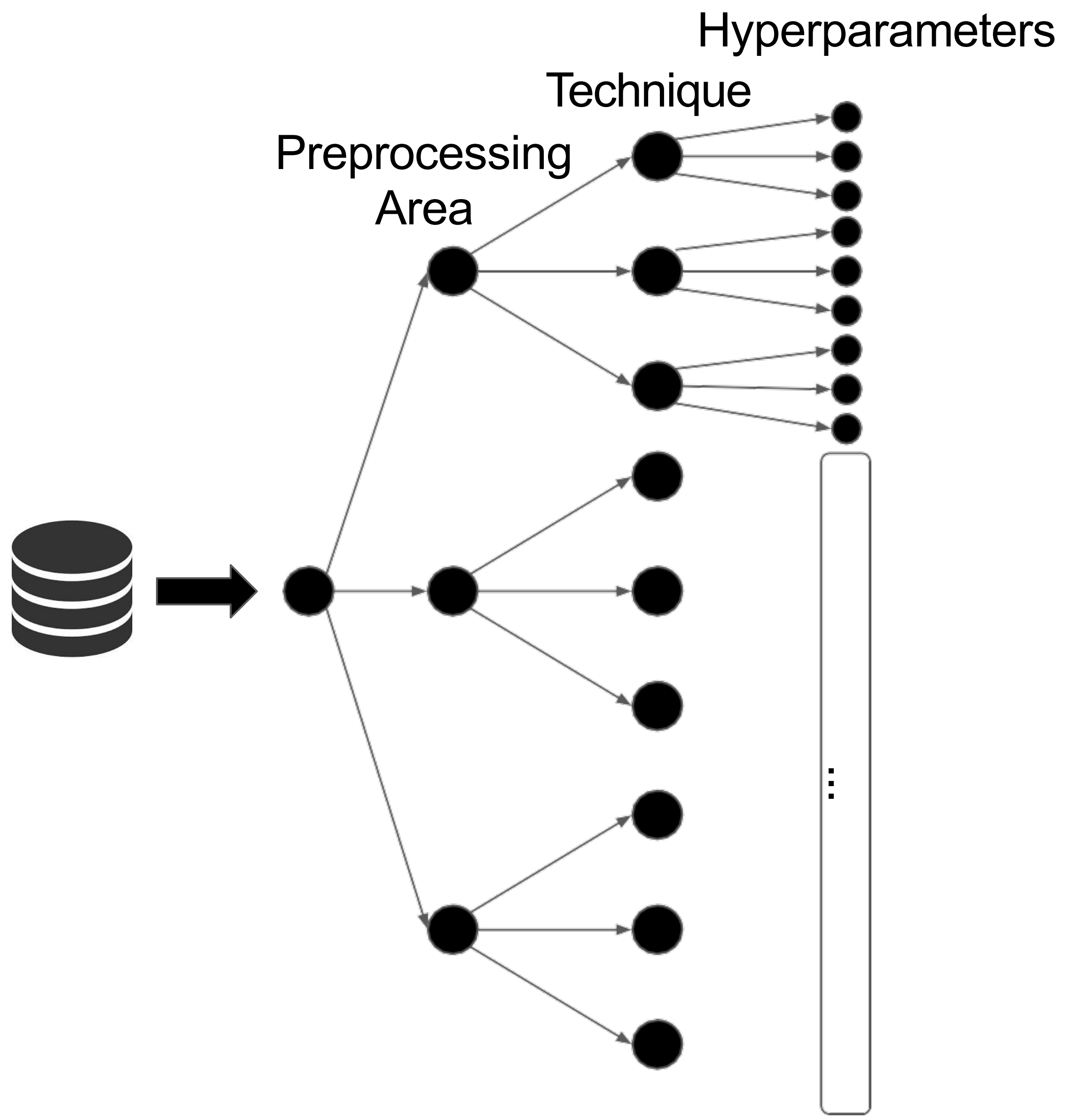


Preprocessing?



→ **We might want more than 1 data preprocessor!**

Complexity of the Preprocessing



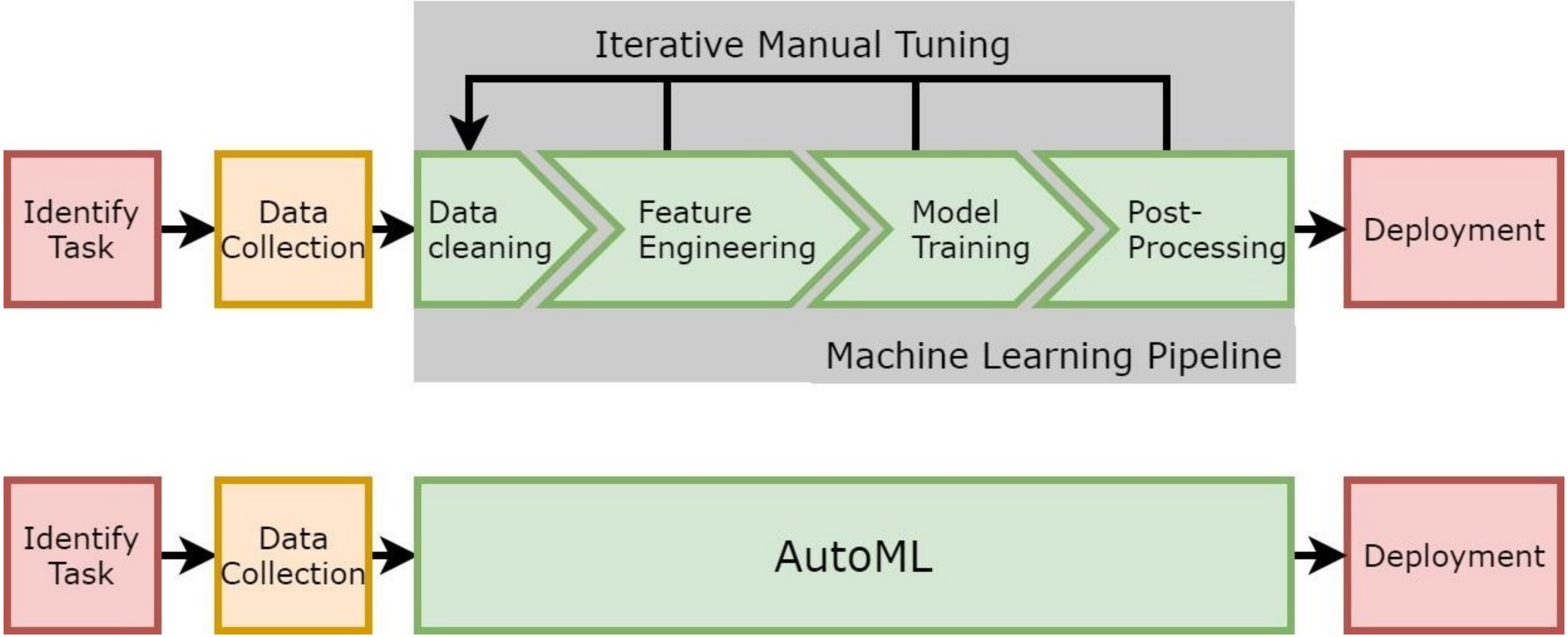
- Naive Assumptions:
  only 3 decisions at each level
- **Possible options**: 3 x 3 x 3 = **27**

- More realistic assumption:
  at least 10 decisions at leach level
- **Possible options:** 10 x 10 x 10 = **1000**

- Choose 3 preprocessors instead of 1
  → 1000 x 1000 x 1000 =
  1 000 000 000

- Still naive!
  → Hyperparameters are often continuous and not discrete
  → ***infinite amount of settings!***

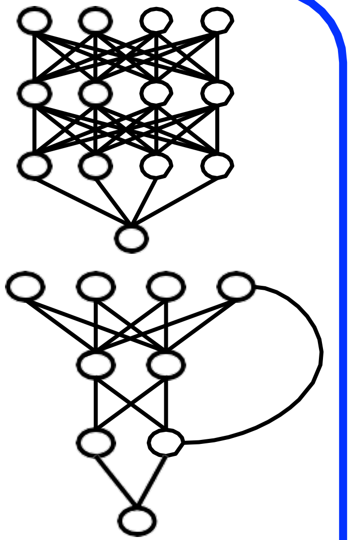
From Manual ML to Automated ML



Design Decisions by AutoML

classifier	#λ
AdaBoost (AB)	4
Bernoulli naïve Bayes	2
decision tree (DT)	4
extreml. rand. trees	5
Gaussian naïve Bayes	-
gradient boosting (GB)	6
kNN	3
LDA	4
linear SVM	4
kernel SVM	7
multinomial naïve Bayes	2
passive aggressive	3
QDA	2
random forest (RF)	5
Linear Class. (SGD)	10

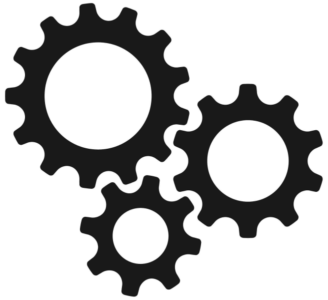
Algorithms



Architecture Design

preprocessor	#λ
extreml. rand. trees prepr.	5
fast ICA	4
feature agglomeration	4
kernel PCA	5
rand. kitchen sinks	2
linear SVM prepr.	3
no preprocessing	-
nystroem sampler	5
PCA	2
polynomial	3
random trees embed.	4
select percentile	2
select rates	3
one-hot encoding	2
imputation	1
balancing	1
rescaling	1

Pre-processing



Hyper-parameters

...

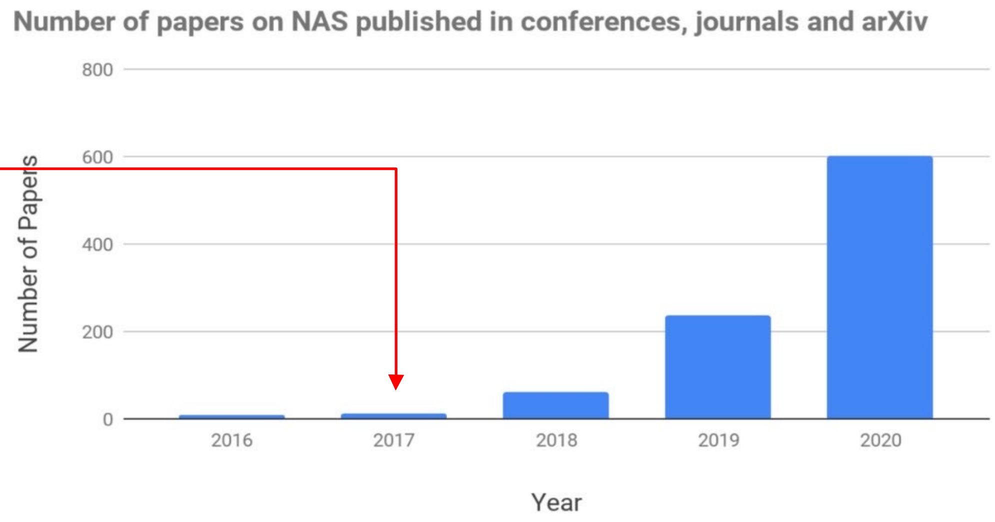
Neural Architecture Search (NAS)

- Find neural architecture A such that deep learning works best for given data
  - Measured by validation error of architecture A with trained weights $w^*(A)$

$$\min_{A \in \mathcal{A}} \mathcal{L}_{\text{val}}(w^*(A), A)$$

$$\text{s.t. } w^*(A) \in \operatorname{argmin}_w \mathcal{L}_{\text{train}}(w, A)$$

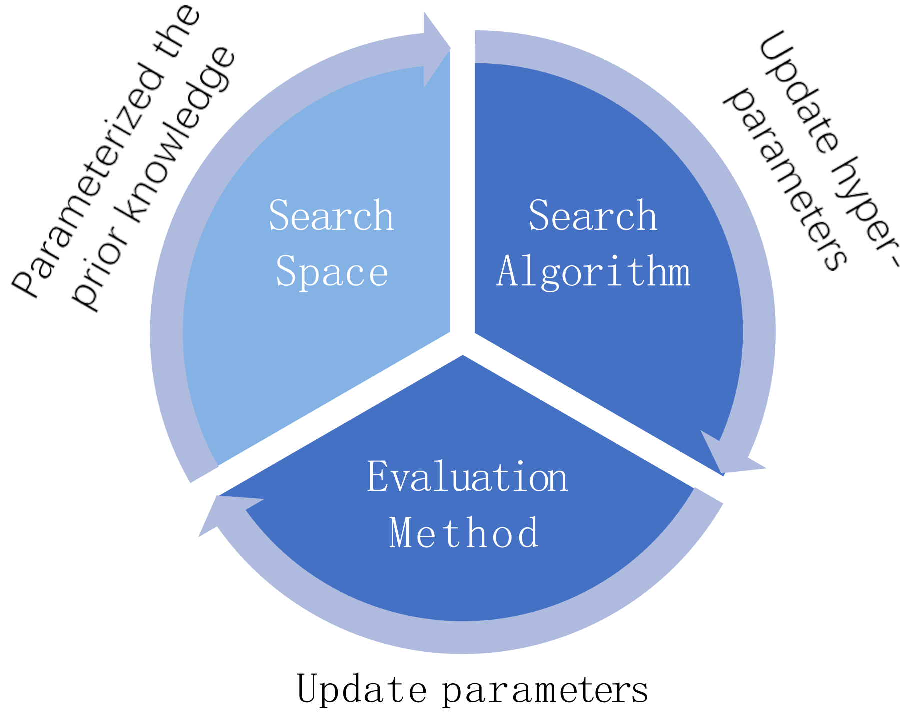
- Famously tackled by reinforcement learning [Zoph & Le, ICLR 2017]
  - 12.800 architectures trained fully
  - 800 GPUs for 2 weeks (about $60.000 USD)



Major Components

- Search Space:
  - A set of operations (e.g. convolution, fully-connected, pooling)
  - how operations can be connected to form valid network architectures

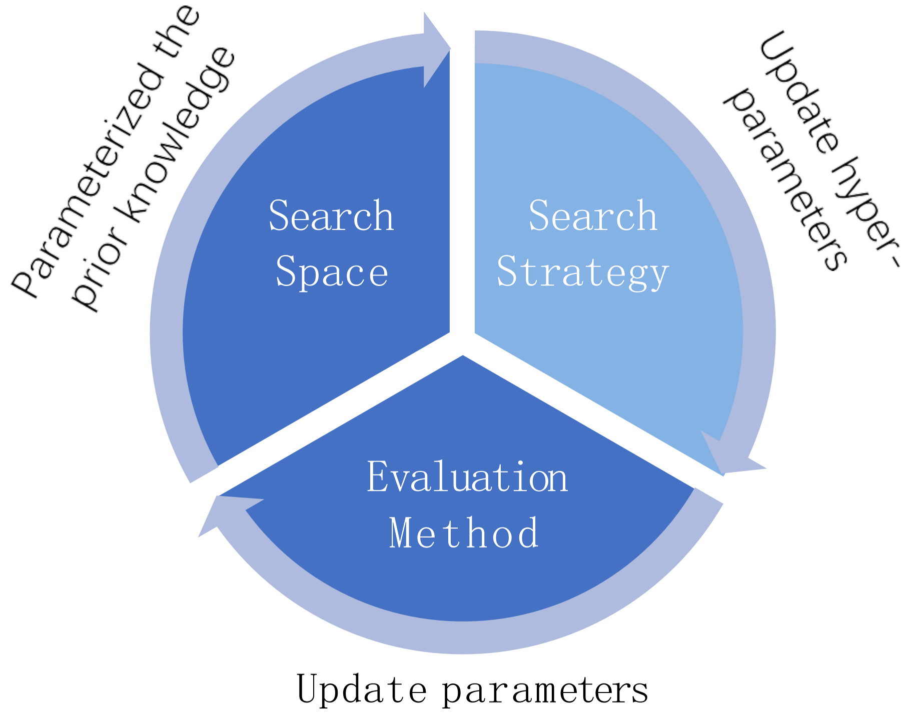
- identity
- 1x7 then 7x1 convolution
- 3x3 average pooling
- 5x5 max pooling
- 1x1 convolution
- 3x3 depthwise-separable conv
- 7x7 depthwise-separable conv
- 1x3 then 3x1 convolution
- 3x3 dilated convolution
- 3x3 max pooling
- 7x7 max pooling
- 3x3 convolution
- 5x5 depthwise-seperable conv



Major Components

- Search Strategy
  - Sampling a population of network architecture candidates (child models)
  - Rewards: child model performance metrics (e.g. high accuracy, low latency)

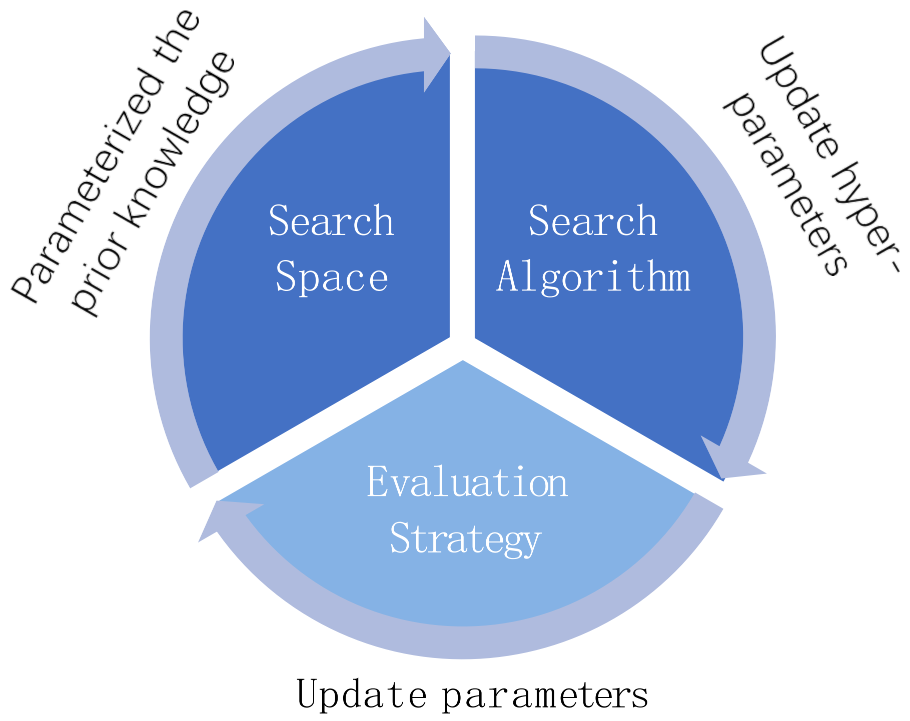
- Algorithms
  - Random Search
  - Reinforcement Learning
  - Gradient descent
  - Evolutionary Algorithms



Major Components

- Evaluation Strategy
  - We need to estimate or predict the performance of child models
  - In order to obtain feedback for the search algorithm to learn

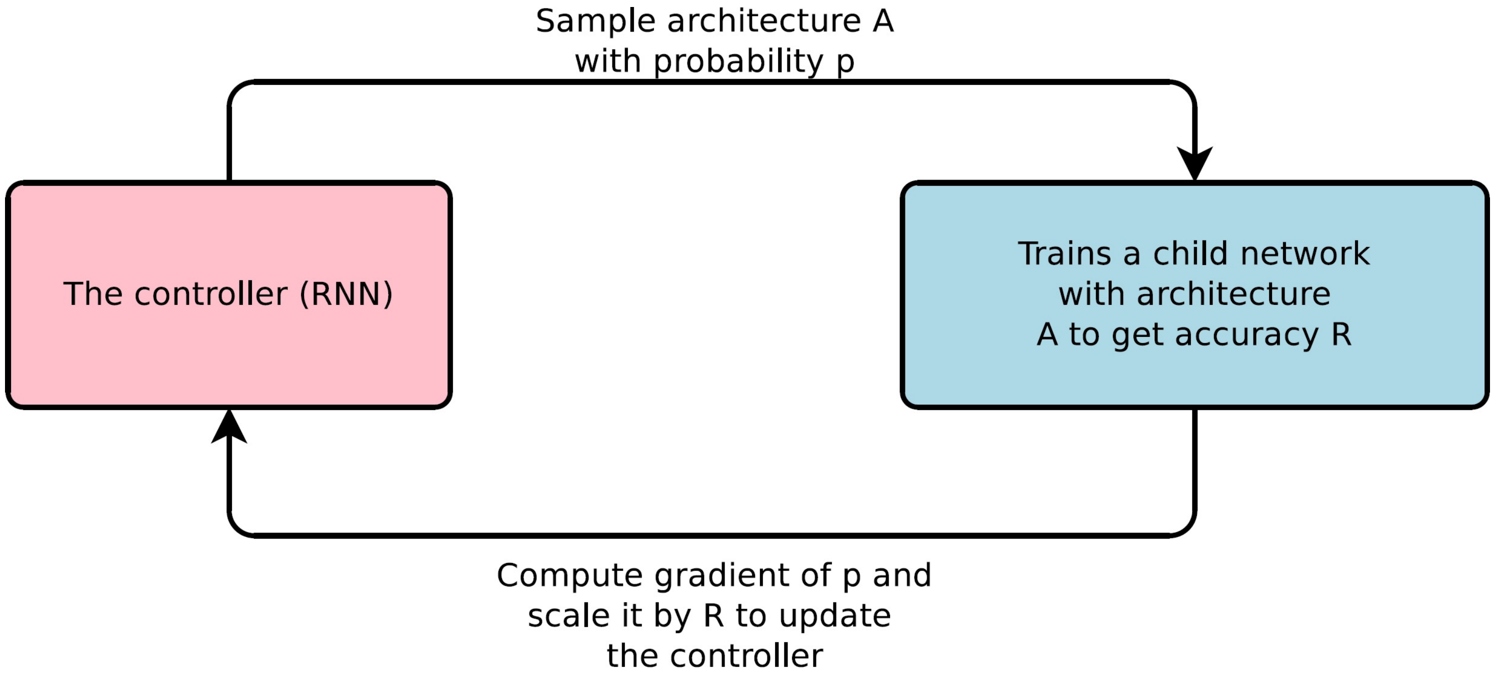
- Methods
  - Training from Scratch
  - Proxy Task Performance
  - Parameter Sharing
  - Prediction-Based



NAS with Reinforcement Learning

- NAS with Reinforcement Learning [Zoph & Le, ICLR 2017]
  - State-of-the-art results for CIFAR-10, Penn Treebank
  - Large computational demands:

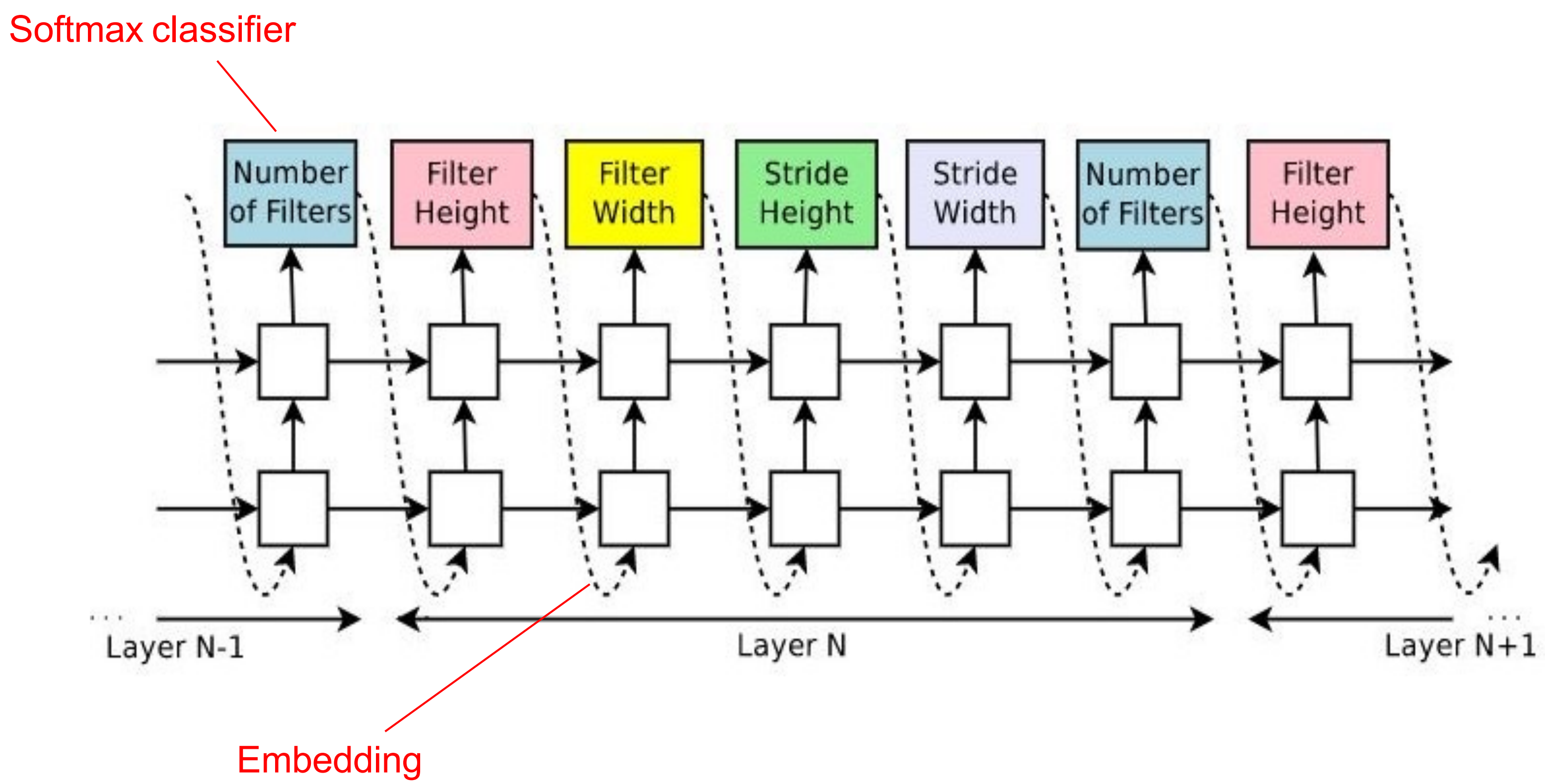
    800 GPUs for 3-4 weeks, 12.800 architectures trained



NAS with Reinforcement Learning

[Zoph & Le, ICLR 2017]

- Architecture of neural network represented as string e.g., ["filter height: 5", "filter width: 3", "# of filters: 24"]
- Controller (RNN) generates string that represents architecture



Training with REINFORCE

Parameters of Controller RNN

Accuracy of architecture on
held-out dataset

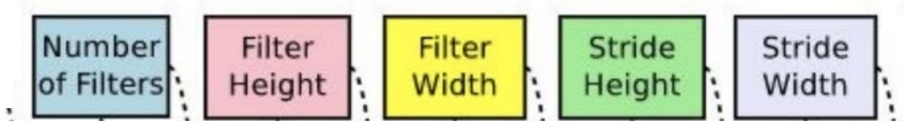
$$J(\theta_c) = E_{P(a_{1:T};\theta_c)}[R]$$

Architecture predicted by the controller RNN
viewed as a sequence of actions

NAS as Hyperparameter Optimization

[Zoph & Le, ICLR 2017]

- Architecture of neural network represented as string e.g., [“filter height: 5”, “filter width: 3”, “# of filters: 24”]
- We can simply treat these as categorical parameters
  - E.g., 25 cat. parameters for each of the 2 cells in [Zoph et al, CVPR 2018]

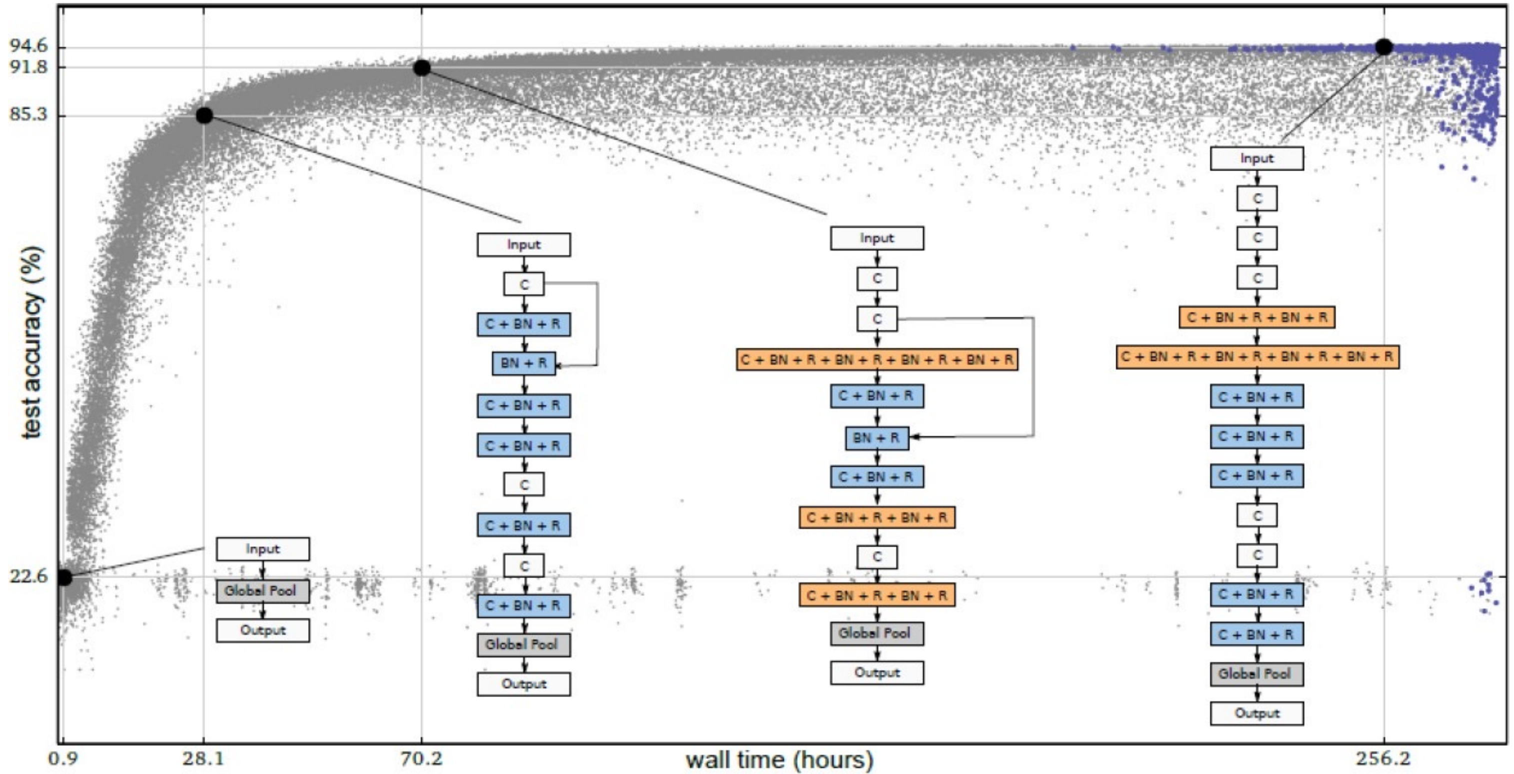


NAS with Evolution

- Neuroevolution

  (already since the 1990s [Angeline et al., 1994; Stanley and Miikkulainen, 2002])
  - Mutation steps, such as adding, changing or removing a layer

    [Real et al., ICML 2017; Miikkulainen et al., arXiv 2017]



Huge Compute of Blackbox Methods

Dataset: CIFAR-10

	Reference	Error (%)	Params (Millions)	GPU Days
RL	Zoph and Le (2017)	3.65	37.4	22,400
RL	Zoph et al. (2018)	3.41	3.3	2,000
EA	Real et al. (2017)	5.40	5.4	2,600
EA	Real et al. (2019)	3.34	3.2	3,150

Going to cell search space

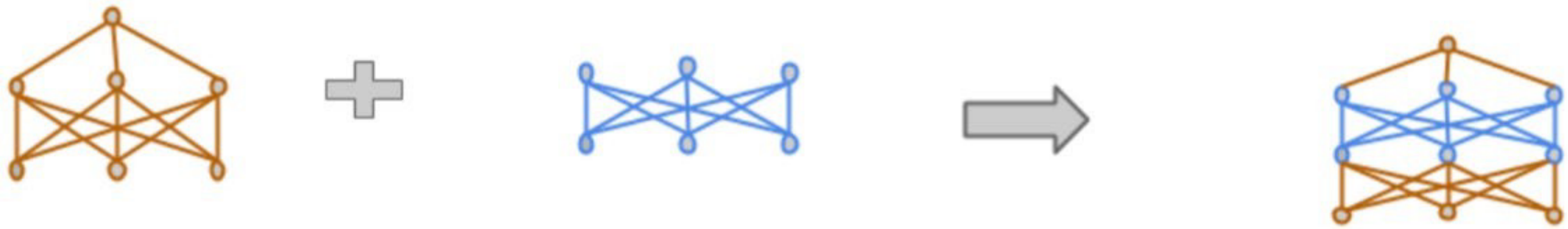
[Wistuba et al., preprint 2019]

Overview of NAS Speedup Techniques

- **Weight Inheritance & Network Morphisms**
  - Local changes in architecture, followed by fine-tuning steps
  - [Cai et al, 2018; Elsken et al, 2017; Cortes et al, 2017; Cai et al, 2018, Elsken et al, 2019]
- **Weight Sharing & One-Shot Models**
  - ENAS [Pham et al, 2018], DARTS [Liu et al, 2019] and many follow-ups
- **Meta-Learning**
  - Learning across datasets
  - To initialize architectural weights of DARTS [Lian et al, 2020; Elsken et al, 2020]
  - Prior for blackbox optimization methods [Wong et al, 2018; Runge et al, 2019; Zimmer et al, 2020]
- **Multi-Fidelity Optimization**
  - Exploit cheaper proxy models for blackbox optimizers, in particular Bayesian optimization
  - [Jamieson & Talwalkar, 2016; Li et al, 2017; Falkner et al, 2018; Zela et al, 2018; White et al, 2021]

Network Morphisms

- Network morphisms [Chen et al., 2016; Wei et al., 2016]
  - Change the network structure, but not the modelled function (i.e., for every input, the network yields the same output
  - as before applying the network morphism)



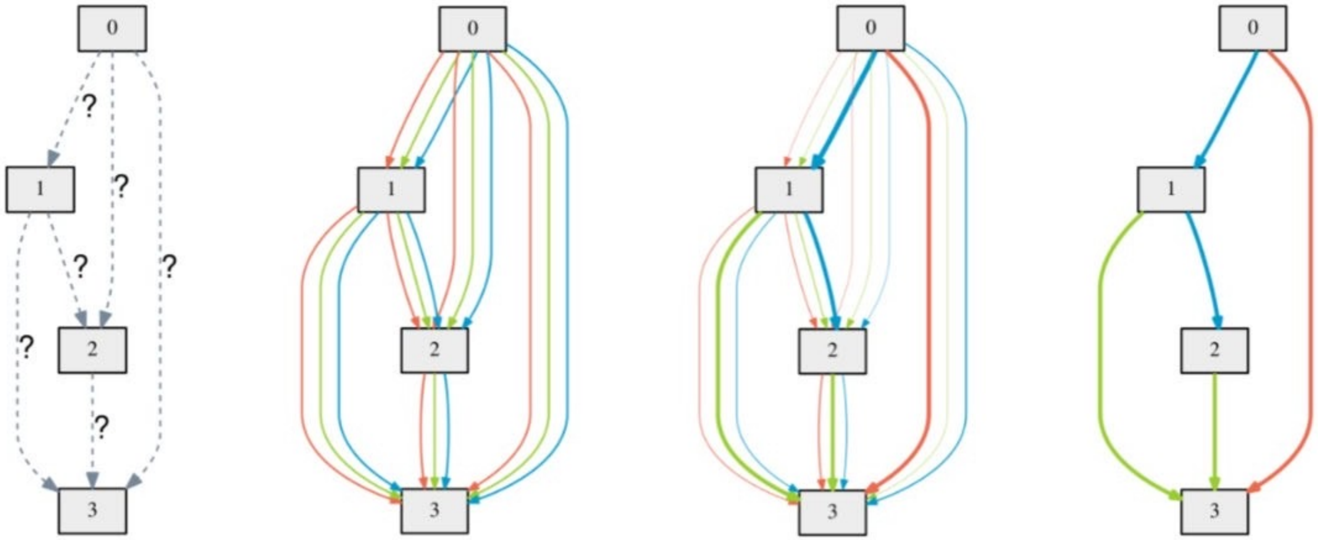
  - Can use this in NAS algorithms as operations to generate new networks
  - Avoids costly training from scratch

Overview of NAS Speedup Techniques

- **Weight Inheritance & Network Morphisms**
  - Local changes in architecture, followed by fine-tuning steps
  - [Cai et al, 2018; Elsken et al, 2017; Cortes et al, 2017; Cai et al, 2018, Elsken et al, 2019]
- **Weight Sharing & One-Shot Models**
  - ENAS [Pham et al, 2018], DARTS [Liu et al, 2019] and many follow-ups
- **Meta-Learning**
  - Learning across datasets
  - To initialize architectural weights of DARTS [Lian et al, 2020; Elsken et al, 2020]
  - Prior for blackbox optimization methods [Wong et al, 2018; Runge et al, 2019; Zimmer et al, 2020]
- **Multi-Fidelity Optimization**
  - Exploit cheaper proxy models for blackbox optimizers, in particular Bayesian optimization
  - [Jamieson & Talwalkar, 2016; Li et al, 2017; Falkner et al, 2018; Zela et al, 2018; White et al, 2021]

DARTS: Differentiable Architecture Search

[Liu et al at ICLR 2019]



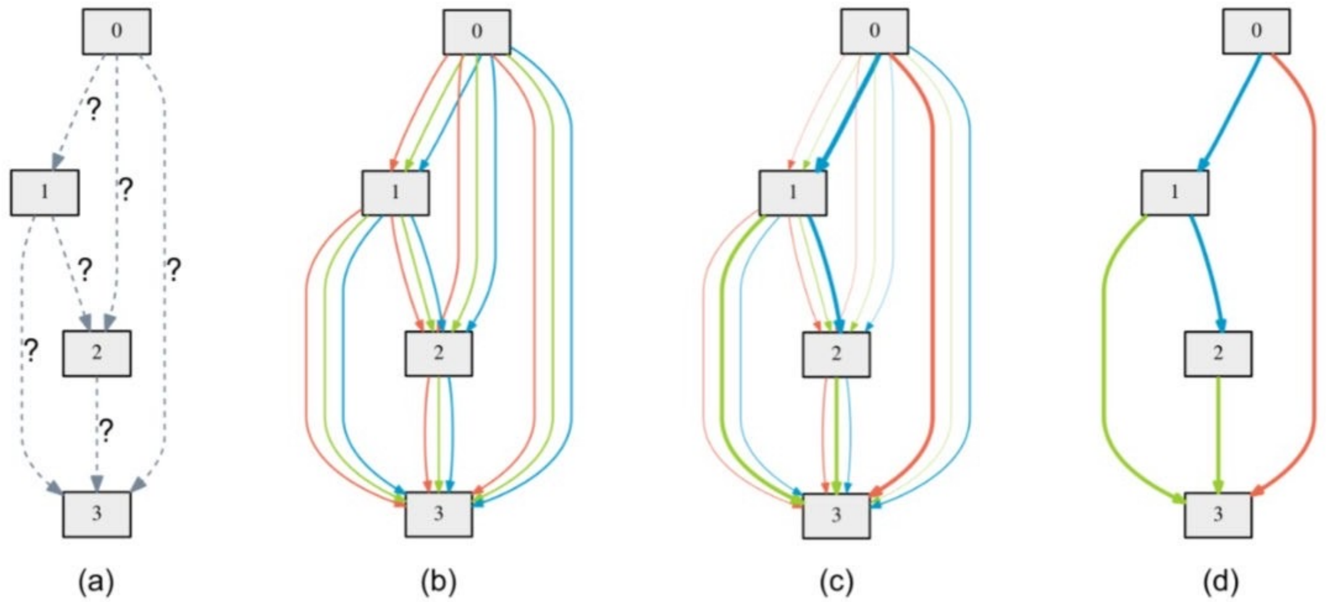
(a) (b) (c) (d)

Candidate operations

- identity
- 1x7 then 7x1 convolution
- 3x3 average pooling
- 5x5 max pooling
- 1x1 convolution
- 3x3 depthwise-separable conv
- 7x7 depthwise-separable conv
- 1x3 then 3x1 convolution
- 3x3 dilated convolution
- 3x3 max pooling
- 7x7 max pooling
- 3x3 convolution
- 5x5 depthwise-seperable conv

DARTS: Differentiable Architecture Search

[Liu et al at ICLR 2019]



- Relax the discrete NAS problem (a->b)
  - One-shot model with continuous architecture weight α for each operator
  - Mixed operator: $\bar{o}^{(i,j)}(x) = \sum_{o \in \mathcal{O}} \frac{\exp(\alpha_o^{(i,j)})}{\sum_{o' \in \mathcal{O}} \exp(\alpha_{o'}^{(i,j)})} o(x)$
- Solve a bi-level optimization problem (c)

$$\min_{\alpha} \quad \mathcal{L}_{val}(w^*(\alpha), \alpha)$$
$$\text{s.t.} \quad w^*(\alpha) = \text{argmin}_w \ \mathcal{L}_{train}(w, \alpha)$$

- In the end, discretize to obtain a single architecture (d)

Table of Contents

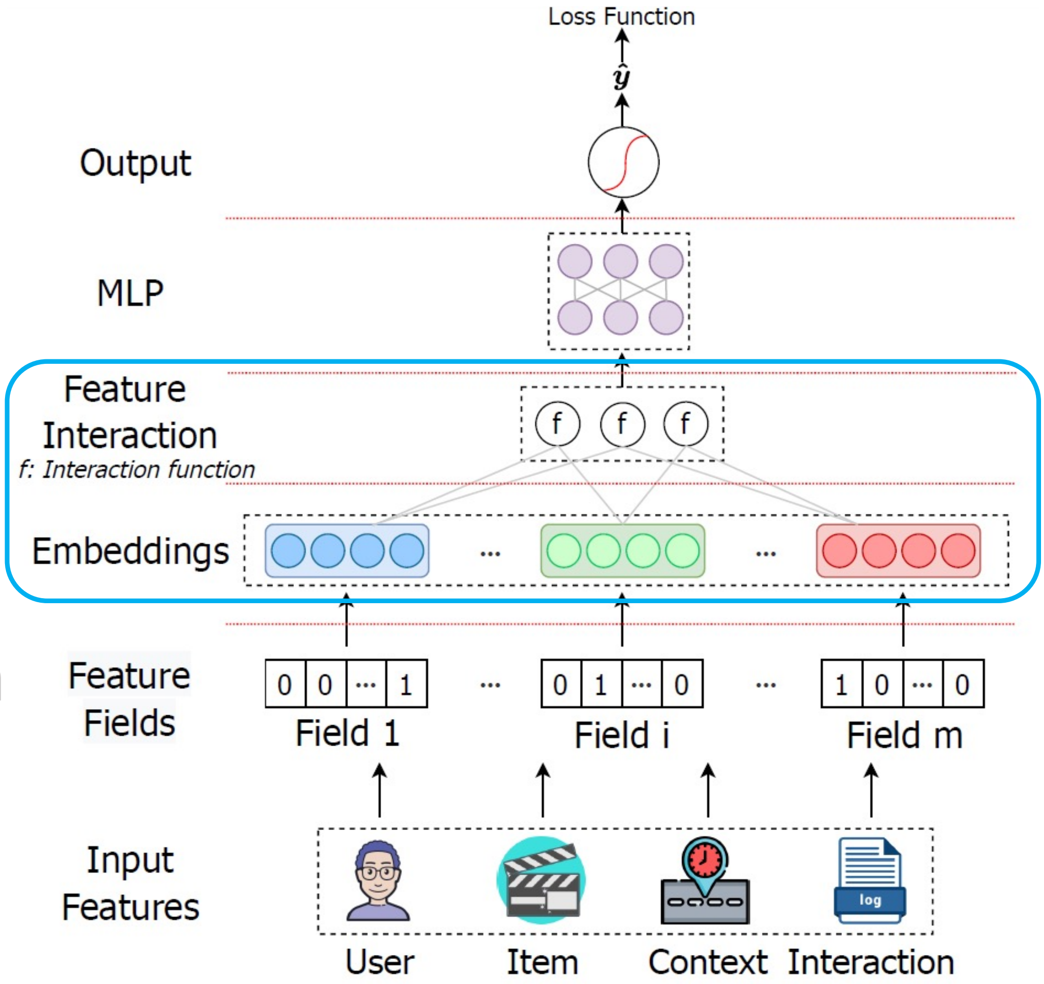


Table of Contents

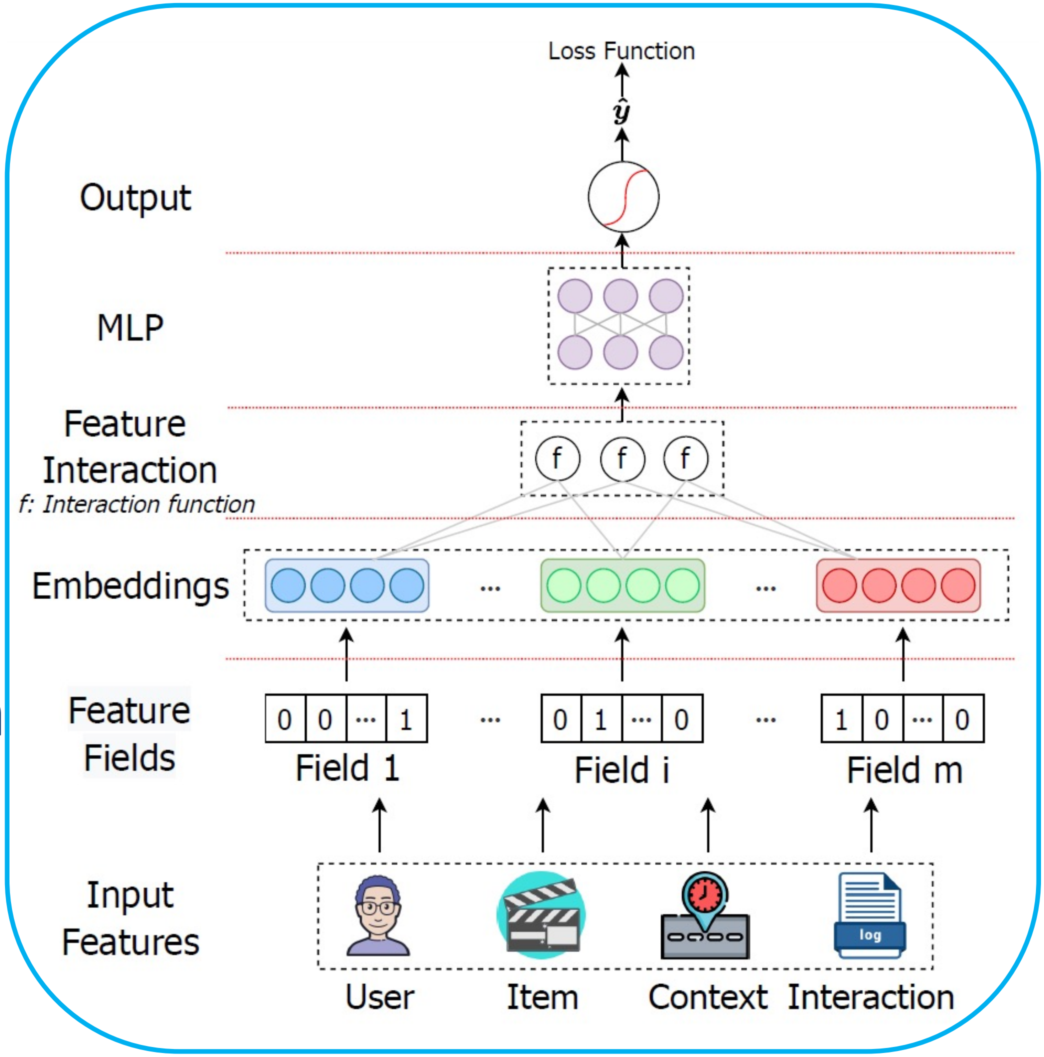
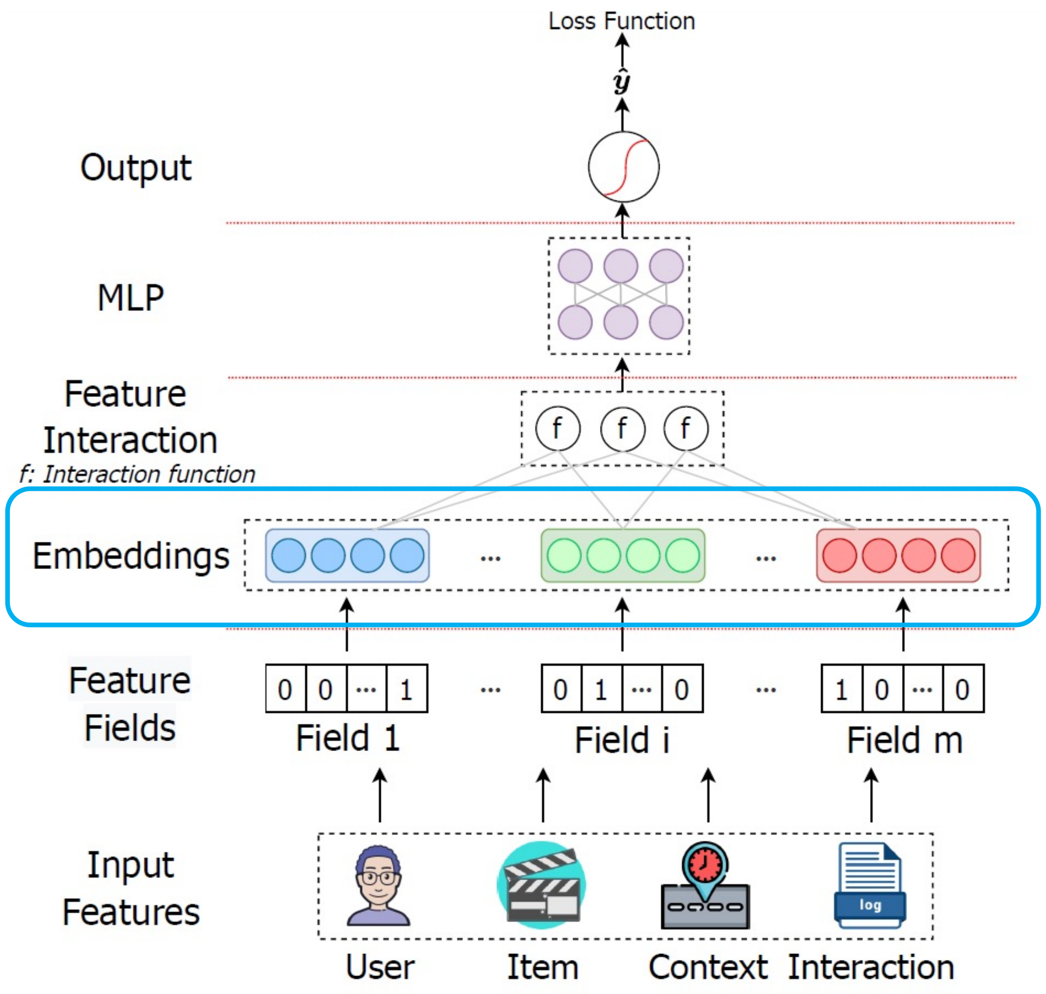


Table of Contents

Background



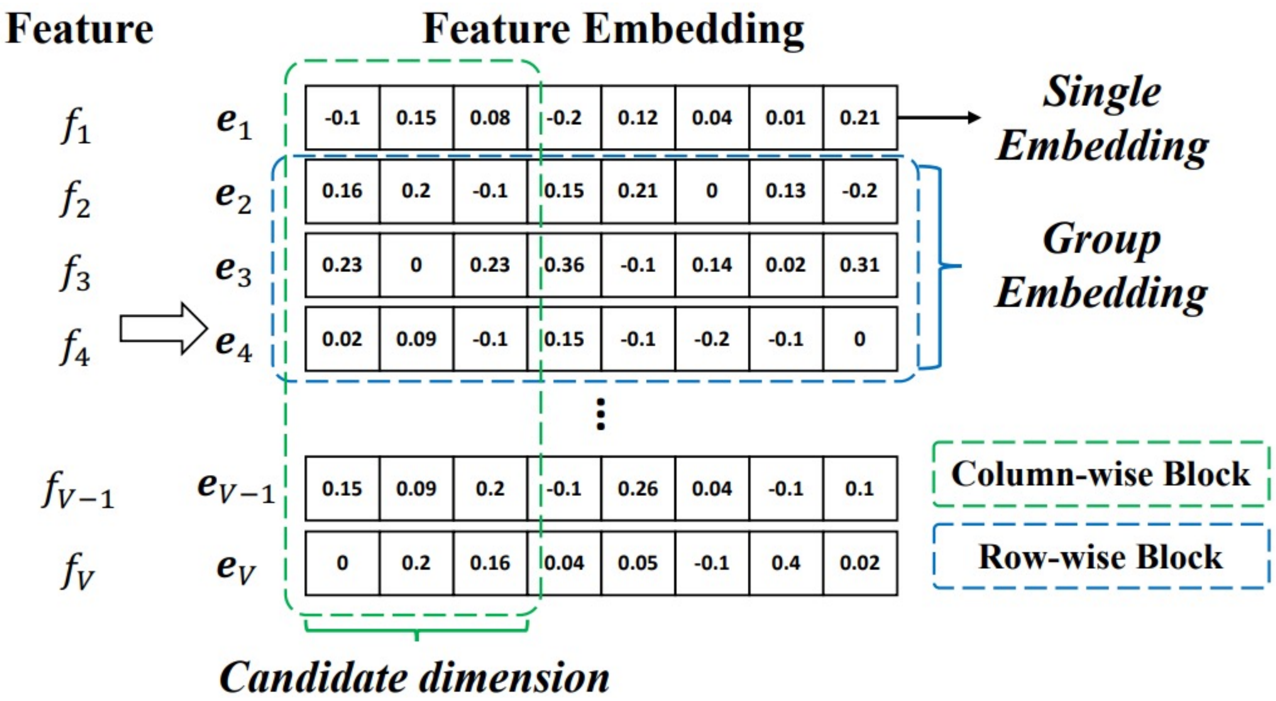
Embedding Table

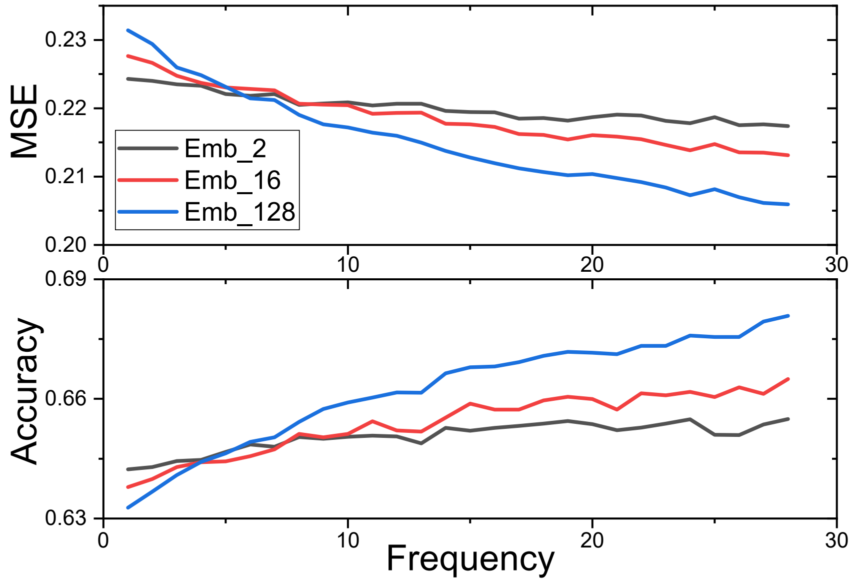
$$E \in R^{V \times d}$$

Number of Feature Values   Embedding Size

- The Embedding layer is used to map **the high-dimensional features** into a **low-dimensional latent space**.
- The **cornerstone of the DRS,** as the number of parameters in DRS is concentrated in the embedding table.

Background

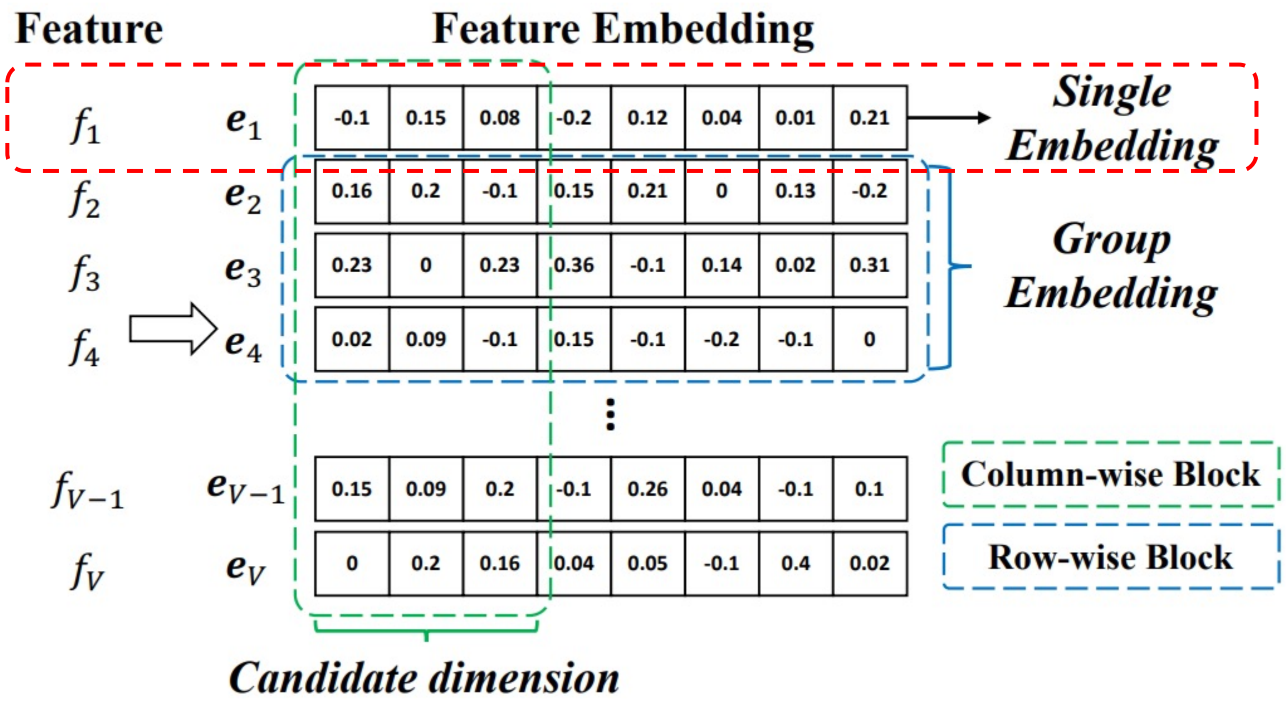




To **improve the prediction accuracy, save storage space and reduce model size**, AutoML-based solutions are proposed for the learning of feature embedding.

1. Single Embedding Search —— search for each feature value
2. Group Embedding Search —— search for a group of feature values

Single Embedding Search



Embedding Table

$$E \in R^{V \times d}$$

Number of Feature Values

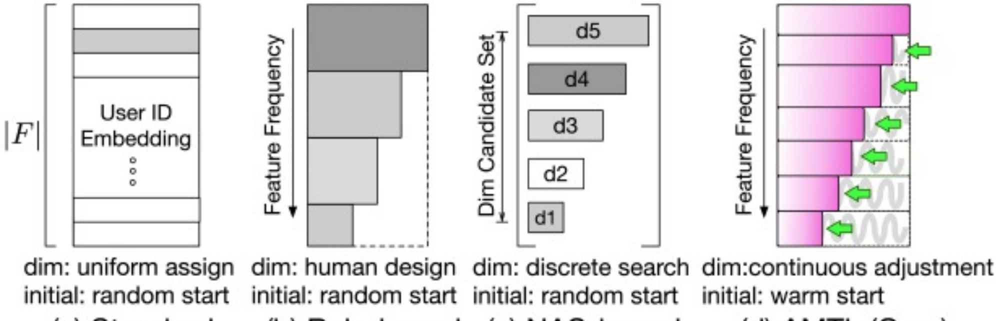
Embedding Size

To improve the prediction accuracy, save storage space and reduce model size, AutoML-based solutions are proposed for the learning of feature embedding.

1. **Single Embedding Search —— search for each feature value**
2. Group Embedding Search —— search for a group of feature values

Single Embedding Search-AMTL

- Search space: d^V (d is the embedding size and V is the vocabulary size)
- **Twins-based architecture** to avoid the unbalanced parameters update problem due to different frequencies.
- The twins-based architecture acts as a frequency-aware policy network to search the optimal dimension for each feature value → relaxed to a continuous space by temperature softmax.



(a) Standard (b) Rule-based (c) NAS-based (d) AMTL (Ours)

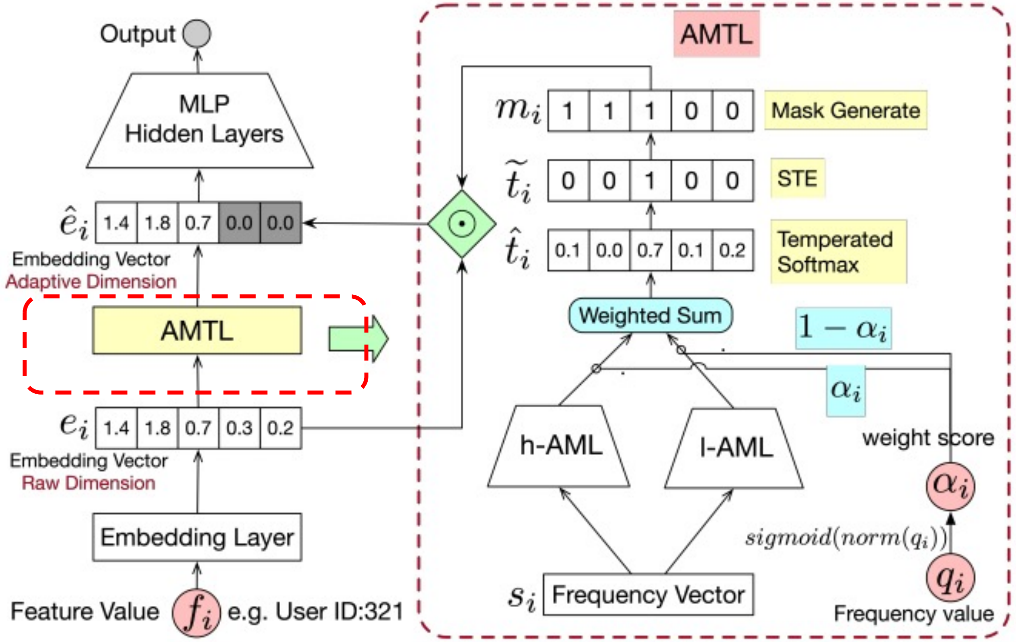


Figure 2: The framework of AMTL.

Learning effective and efficient embedding via an adaptively-masked twins-based layer. CIKM 2021.

Single Embedding Search-PEP

- Pruning-based Solution by enforcing **column-wise sparsity** on the embedding table with L_0 normalization.
- Search Space: 2^{Vd} (d is the embedding size and V is the vocabulary size)

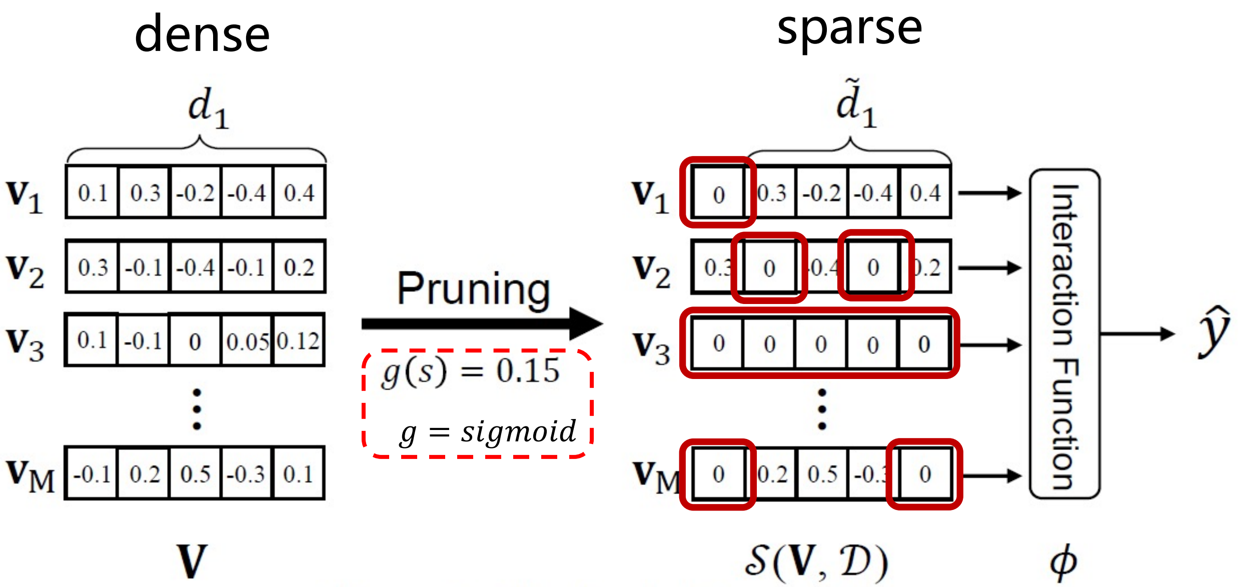


Figure 1: The basic idea of PEP.

$\min \mathcal{L}, s.t. ||\mathbf{V}||_0 \leq k,$ NP-hard

Soft threshold re-parameterization :

$$\hat{\mathbf{V}} = \mathcal{S}(\mathbf{V}, s) = sign(\mathbf{V})\text{ReLU}(|\mathbf{V}| - g(s)),$$

$$\min \mathcal{L}(\mathcal{S}(\mathbf{V}, s), \Theta, \mathcal{D}).$$

PEP: Learnable Embedding Sizes for Recommender Systems. ICLR 2021.

Single Embedding Search

- The search space of PEP and AMTL is highly related with the embedding size d.
- To reduce the search space, AutoEmb and ESPAN divide the embedding dimension into several **column-wise sub-dimensions**.

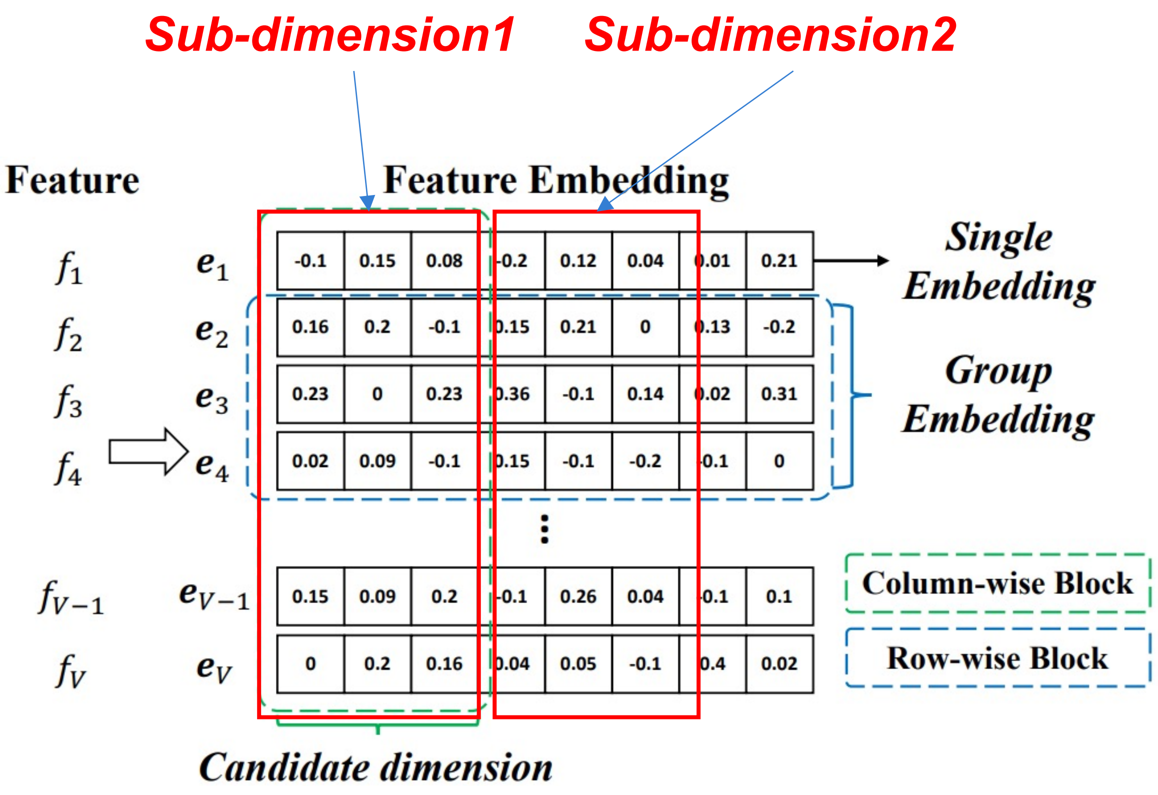
Solution 1:
column-wise sub-dimensions

Embedding Table

$$\mathrm{E} \in R^{V \times d}$$

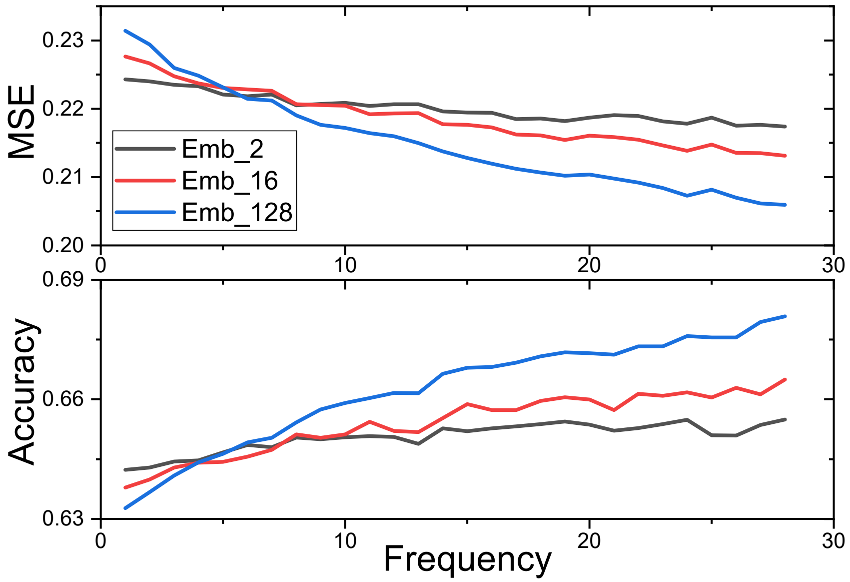
Number of Feature Values

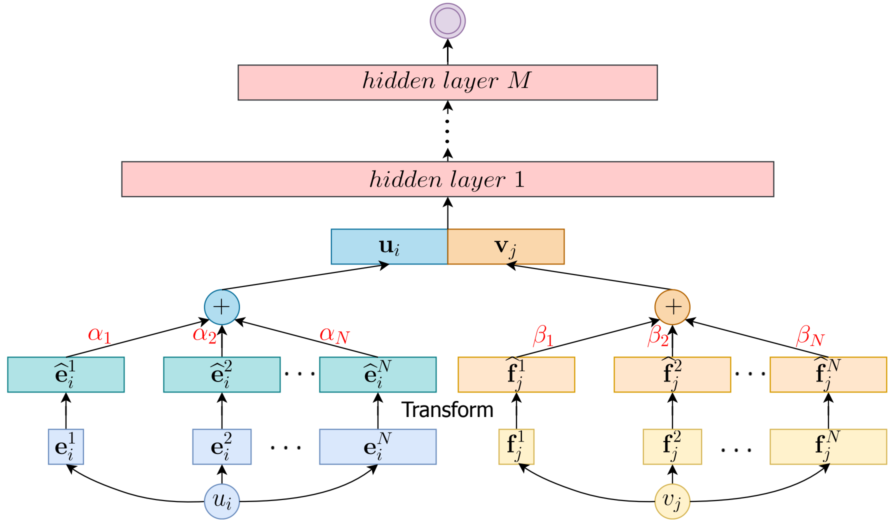
Embedding Size



Single Embedding Search-AutoEmb & ESPAN

- Reduce the search space by dividing the embedding dimension into several **candidate sub-dimensions**
- Embedding dimension often determines the capacity to encode information.
- Dynamically search the embedding sizes for different **users and items**
  - Optimal recommendation quality all the time
  - More efficient in memory



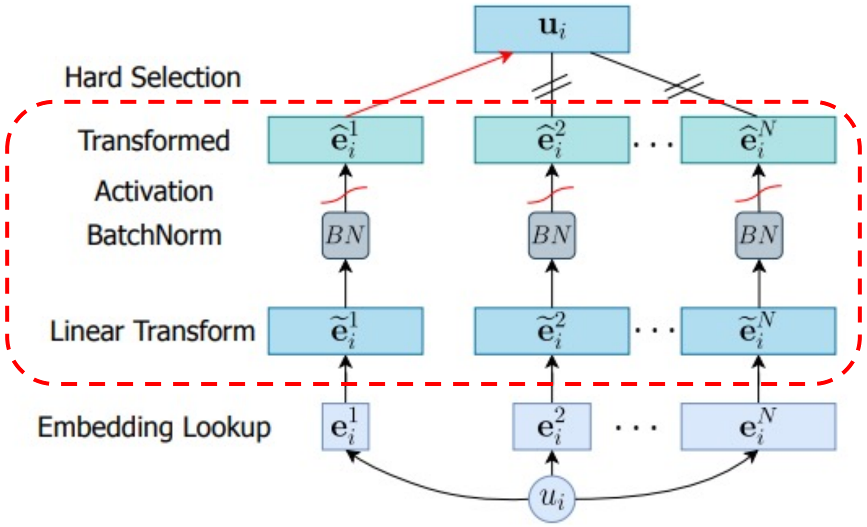


AutoEmb & ESPAN

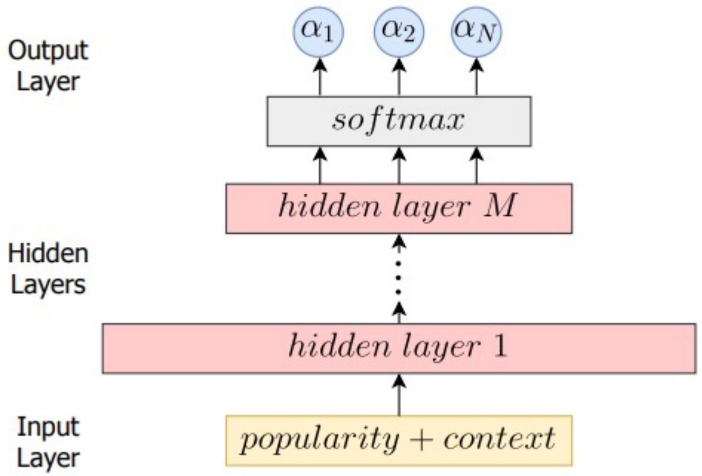
ESAPN: Automated Embedding Size Search in Deep Recommender Systems. SIGIR 2020.

Single Embedding Search-AutoEmb

- Search Space: From d^V to a^V (d is the embedding size and V is the vocabulary size, and a is **the number of sub-dimensions** for each feature)
- Two **controller networks** to decide the embedding sizes for users and items via end-to-end differentiable soft selection.
- Sum over the candidate subdimensions with learnable weights. **(Soft Selection)**



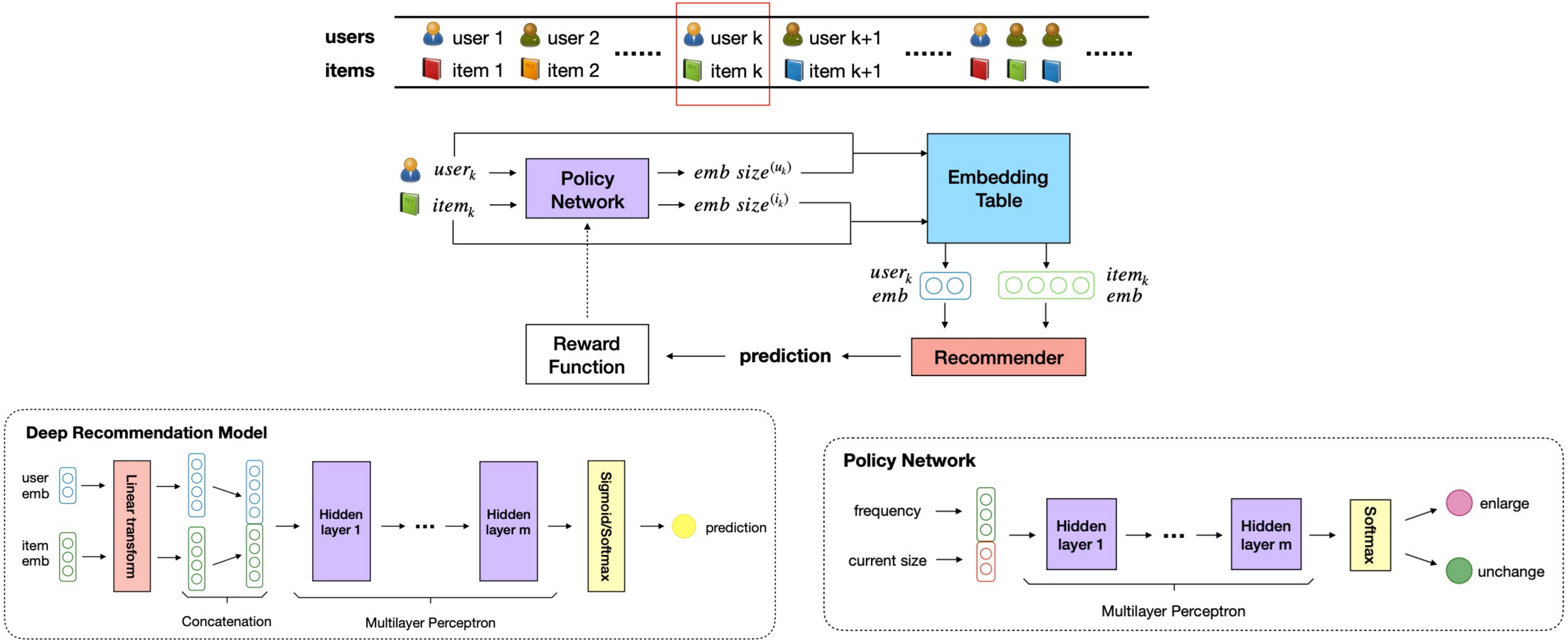
AutoEmb



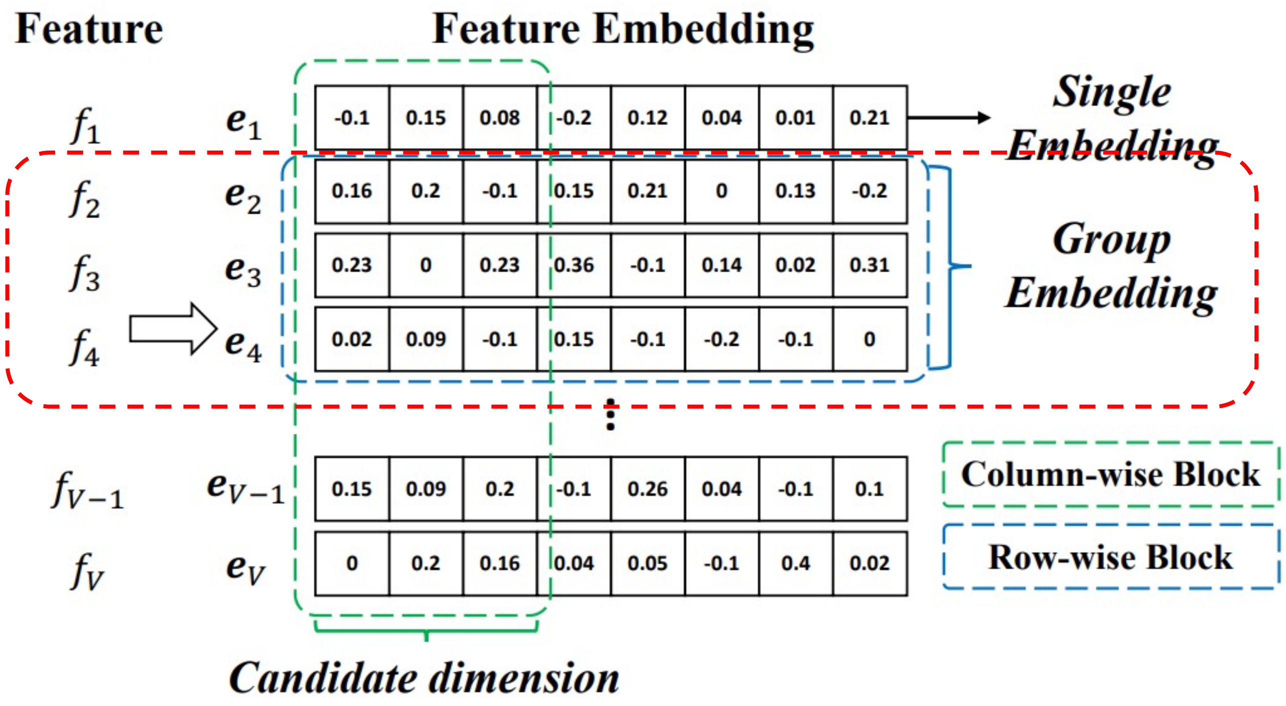
Controller network

Single Embedding Search-ESPAN

- Two Components
  - Deep recommendation model
  - Embedding Size Adjustment Policy Network (ESAPN) - RL **(Hard Selection)**



Group Embedding Search



Embedding Table

$$E \in R^{V \times d}$$

Number of Feature Values

Embedding Size

To improve the prediction accuracy, save storage space and reduce model size, AutoML-based solutions are proposed for the learning of feature embedding.

1. Single Embedding Search —— search for each feature value
2. **Group Embedding Search —— search for a group of feature values**

Group Embedding Search

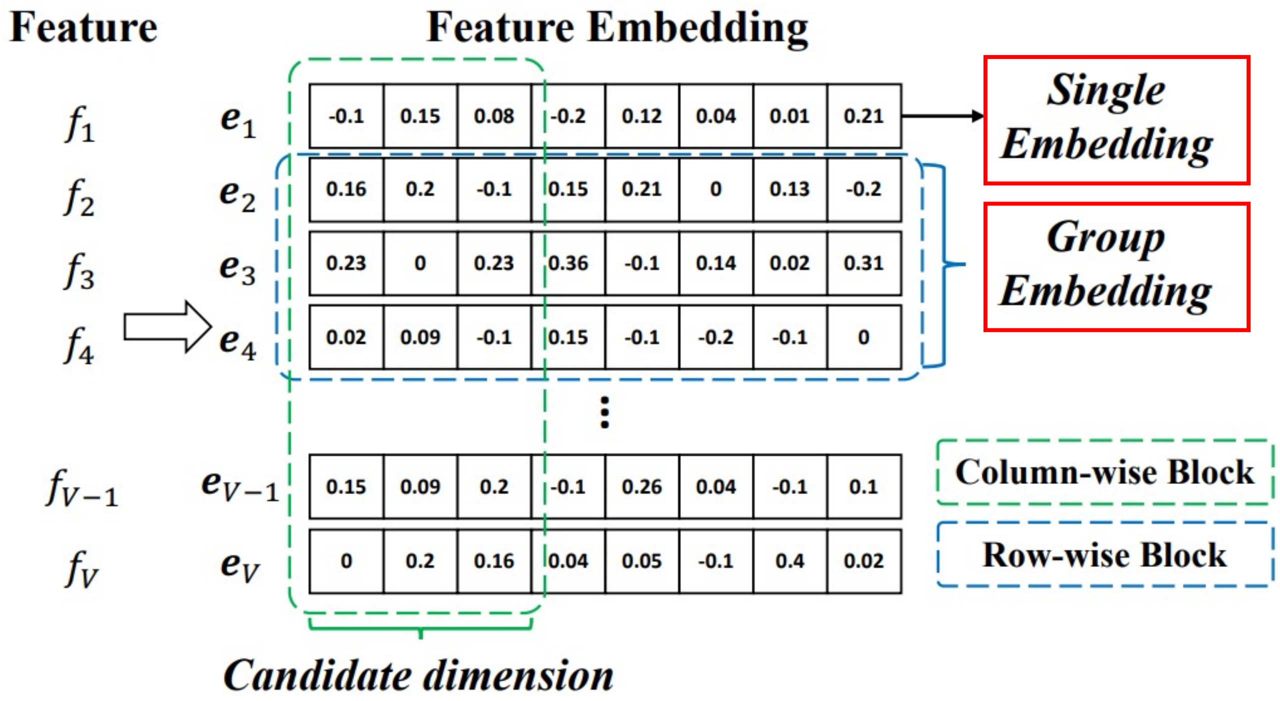
- AutoEmb and ESAPN shrink the search space by dividing the embedding dimension into candidate **column-wise sub-dimensions**.
- **Group** the feature values of a field based on some indicators (e.g., frequencies) and assign a **row-wise group embedding dimension** for all the values within the group.

Solution 2:
row-wise group embedding dimension

Embedding Table

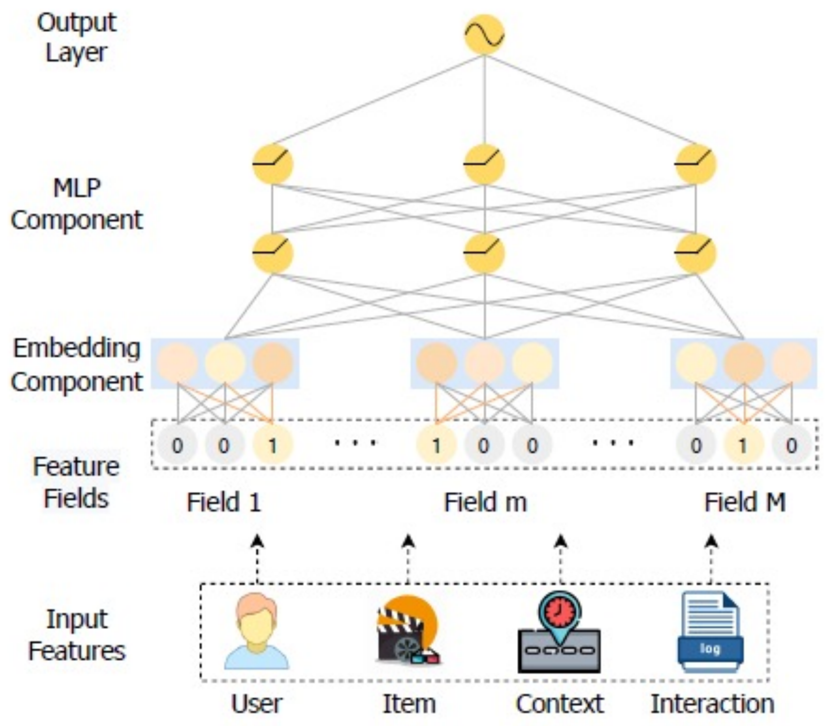
$$\mathrm{E} \in R^{V \times d}$$

Number of Feature Values   Embedding Size



Group Embedding Search-AutoDim

- Pre-defines several candidate sub-dimensions like AutoEmb.
- Setting the number of groups b = 1 and searching a global embedding dimension for **all the feature values of the field**.
- Search Space: From a^V to a^{m} (where d is the embedding size and V is the vocabulary size, and a is **the number of sub-dimensions** for each feature, m is **the number of feature fields**)

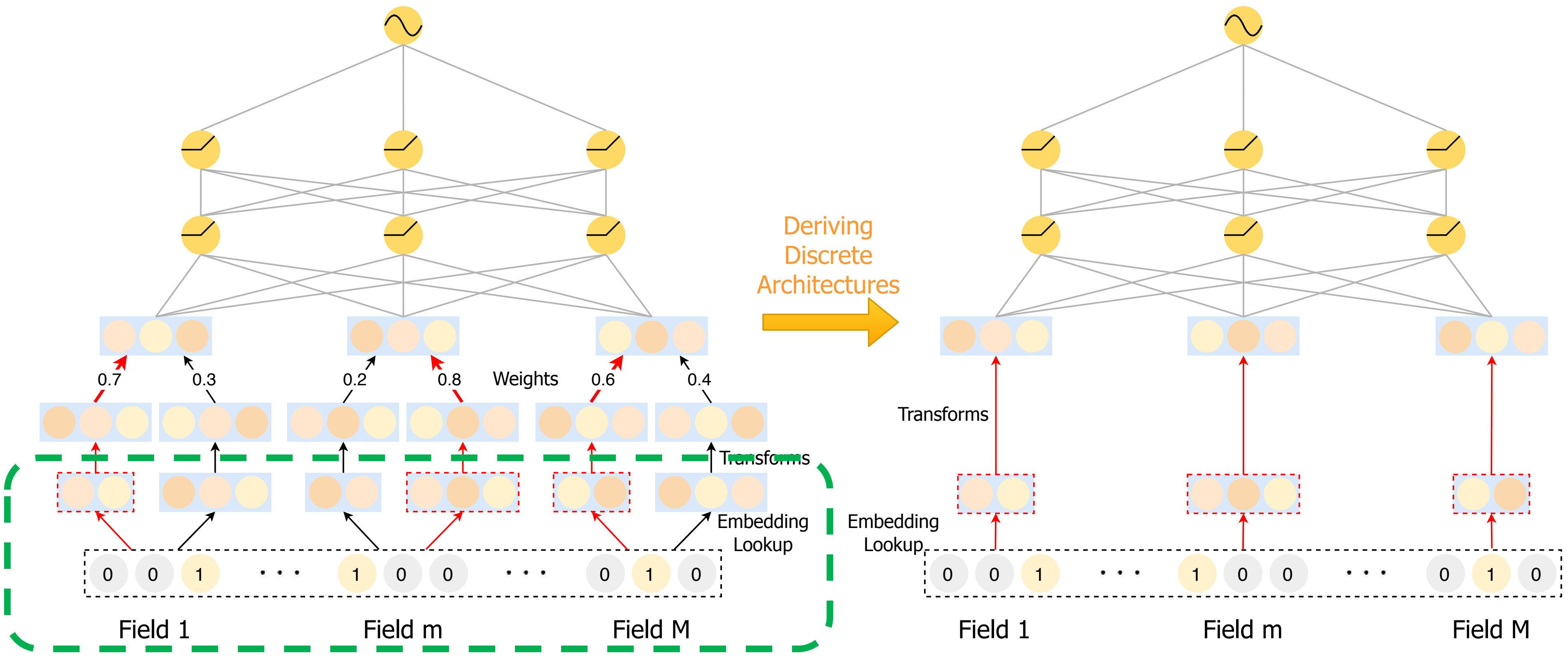


Goal:
Selecting embedding dimensions to **different feature fields** automatically in a data-driven manner.

AutoDim: Field-aware Embedding Dimension Search in Recommender Systems. WWW 2021.

Group Embedding Search-AutoDim

Two-stage framework

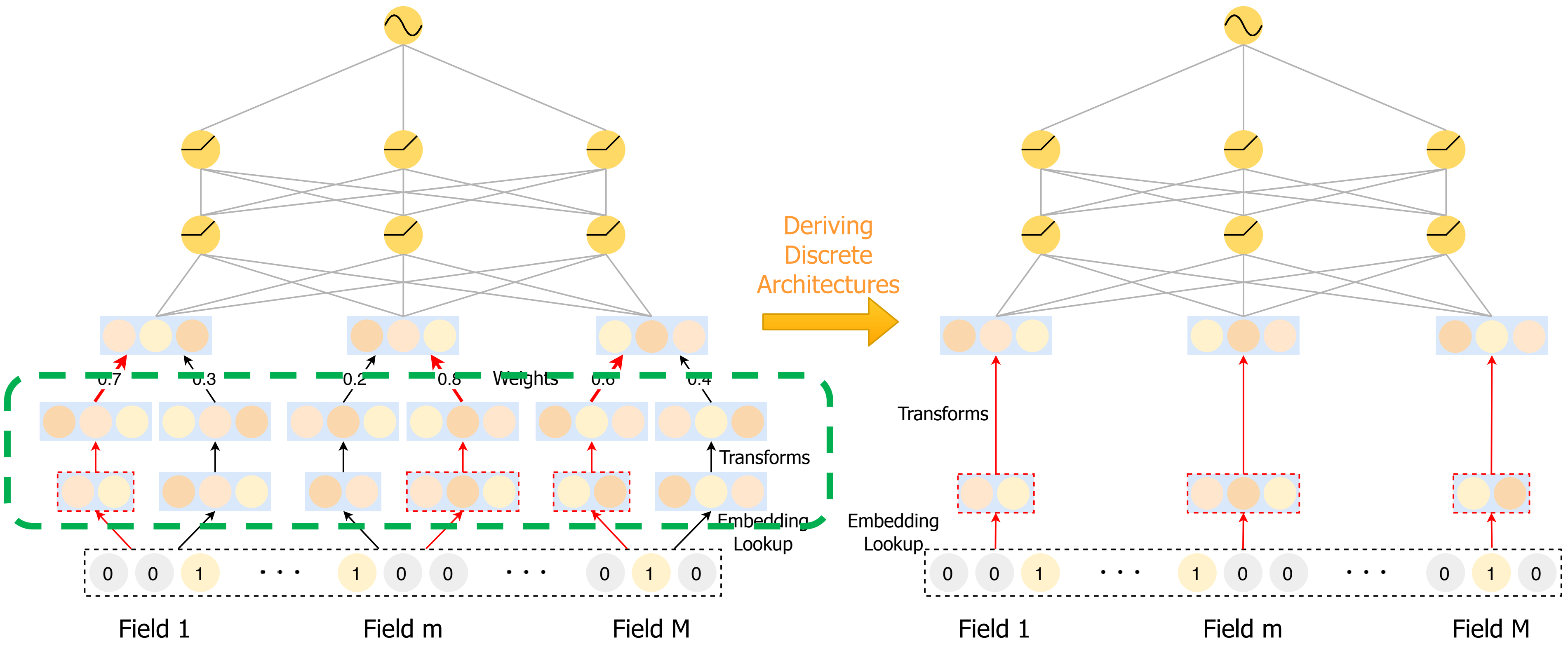


(a) Dimensionality Search (b) Parameter Re-training

AutoDim searches dimensions for feature fields in a soft and continuous fashion via **Gumbel Softmax**, reducing to a smaller serach space: 5 candiate for each feature field.

Group Embedding Search-AutoDim

Two-stage framework



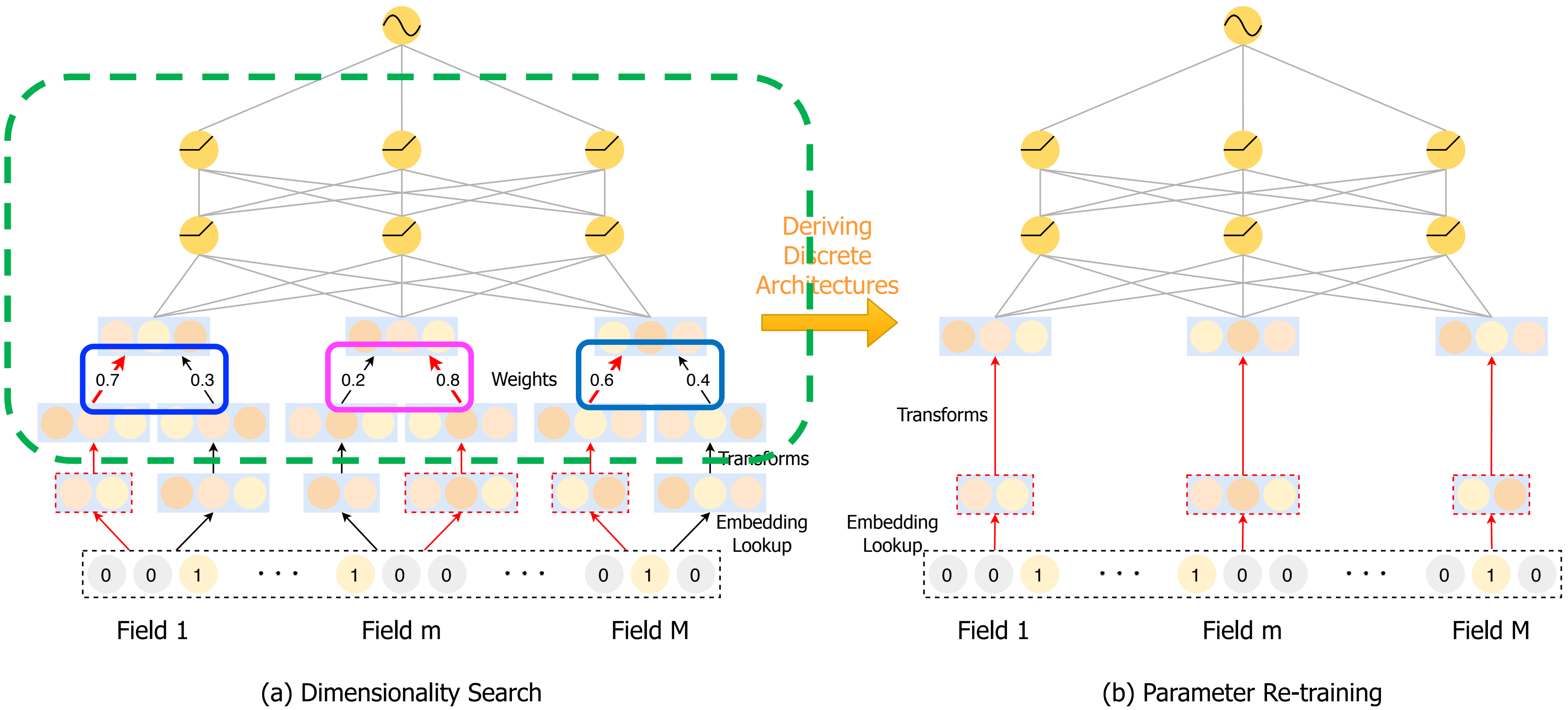
(a) Dimensionality Search

(b) Parameter Re-training

AutoDim searches dimensions for feature fields in a soft and continuous fashion via **Gumbel Softmax**, reducing to a smaller serach space: 5 candiate for each feature field.

Group Embedding Search-AutoDim

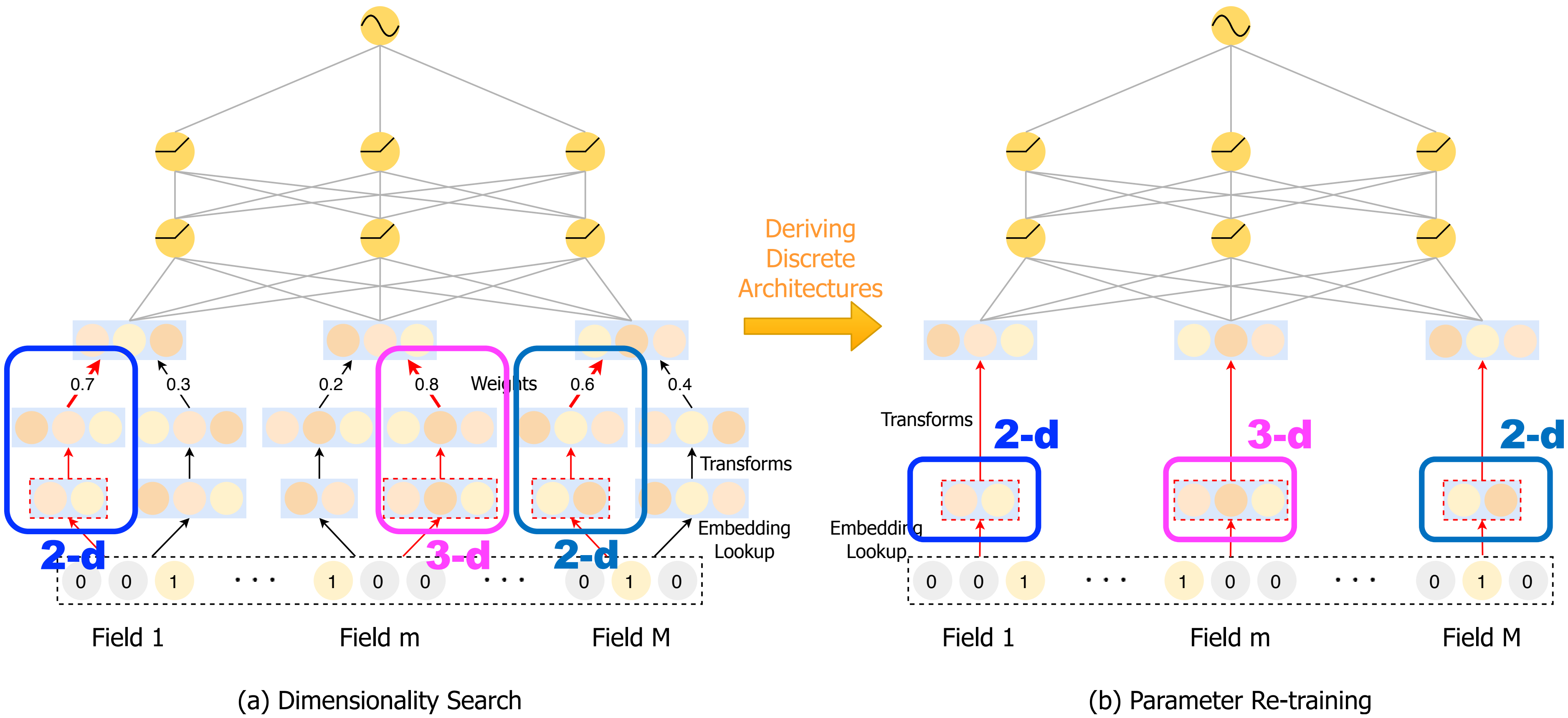
Two-stage framework



AutoDim searches dimensions for feature fields in a soft and continuous fashion via **Gumbel Softmax**, reducing to a smaller serach space: 5 candiate for each feature field.

Group Embedding Search-AutoDim

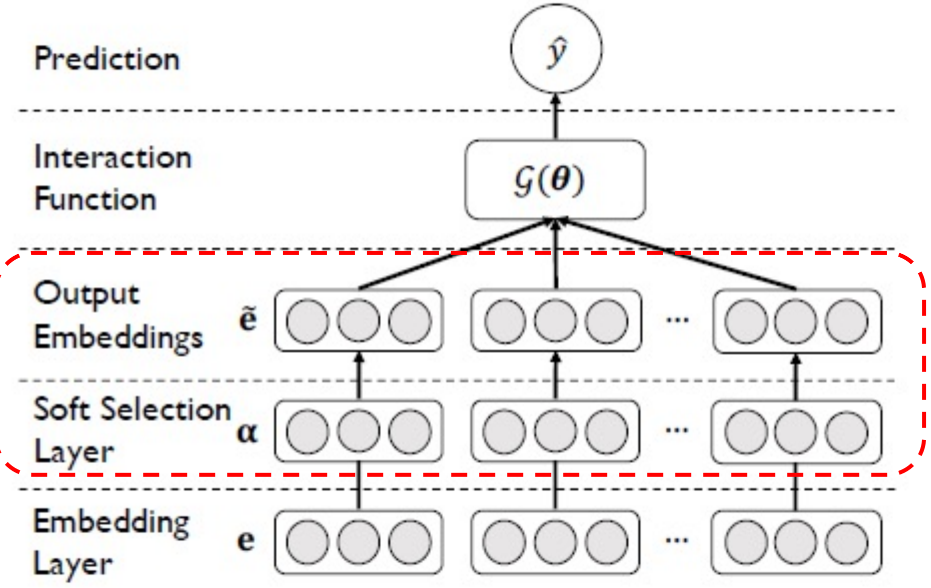
Two-stage framework



AutoDim searches dimensions for feature fields in a soft and continuous fashion via **Gumbel Softmax**, reducing to a smaller serach space: 5 candiate for each feature field.

Group Embedding Search-DNIS

- Split the features into **multi-groups** based on the *feature frequencies or clustering*.
- Search space: from 2^{Vd} into 2^{bd} (where b is **the number of groups**)
- Search for mixed feature embedding dimensions in a more flexible space through continuous relaxation and differentiable optimization.



(b) Model structure.

Search stage: $\tilde{\mathbf{e}}_i = \mathbf{e}_i \odot \boldsymbol{\alpha}_{l*}$

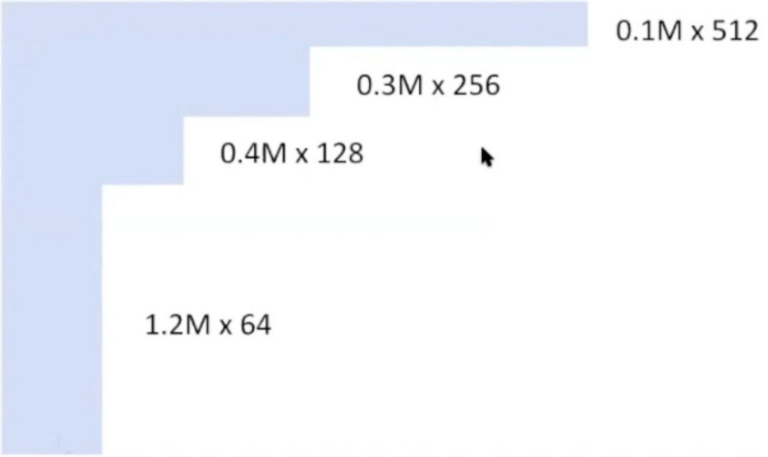
Derive stage: $\tilde{\mathbf{E}}_{i,j} = \begin{cases} 0, & \text{if } |\tilde{\mathbf{E}}_{i,j}| < \epsilon \\ \tilde{\mathbf{E}}_{i,j}, & \text{otherwise} \end{cases}$

DNIS: Differentiable Neural Input Search for Recommender Systems. In Arxiv.

Group Embedding Search-NIS

- Input component assigns embedding vectors to each item of these discrete features, which dominate both the size and the inductive bias of the model.
- The vocabulary and embedding sizes for discrete features are often selected heuristically.

Solution 3:
column-wise sub-dimensions
row-wise group embedding dimension



Head Feature
- More data, more information
- Needing larger embedding size

Tail Feature
- Less data, less information
- Small embedding size is enough

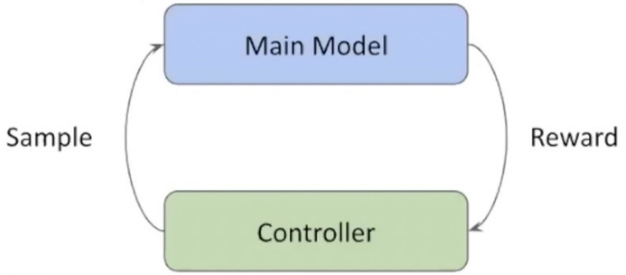
NIS: Neural Input Search for Large Scale Recommendation Models. KDD 2020.

Group Embedding Search-NIS

RL-based AutoML approach

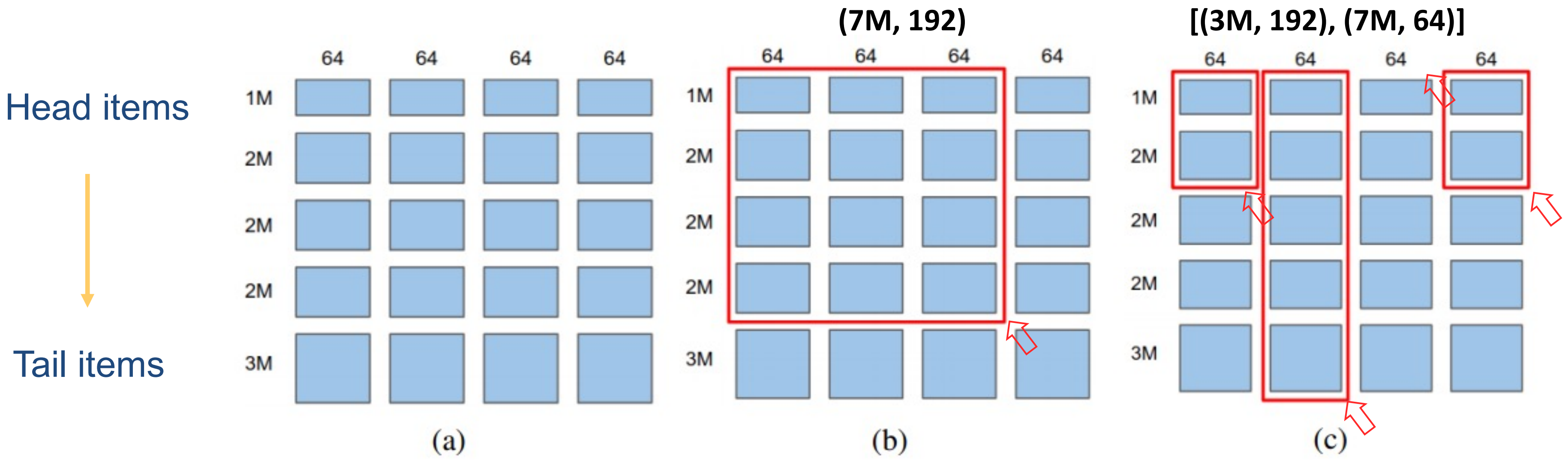
- Main model is the deep recommendation model
- Controller learns to sample embedding dimensions that generate higher reward.
- Reward:

$$R = R_Q - \lambda * C_M$$



$$\sum_{F \in \mathcal{F}} v_F \times d_F \leq \mathcal{C}$$

$$\sum_{F \in \mathcal{F}} \sum_{i=1}^{M_F} v_{F_i} \times d_{F_i} \leq \mathcal{C}$$



Single-size Embedding (SE) **Multi-size Embedding (ME)**

Embedding Blocks: discretizing an embedding matrix of size v × d into S × T sub-matrices

Numerical Embedding-AutoDis

Existing methods for numerical feature representation have some **limitations**:

1. Category 1: No Embedding

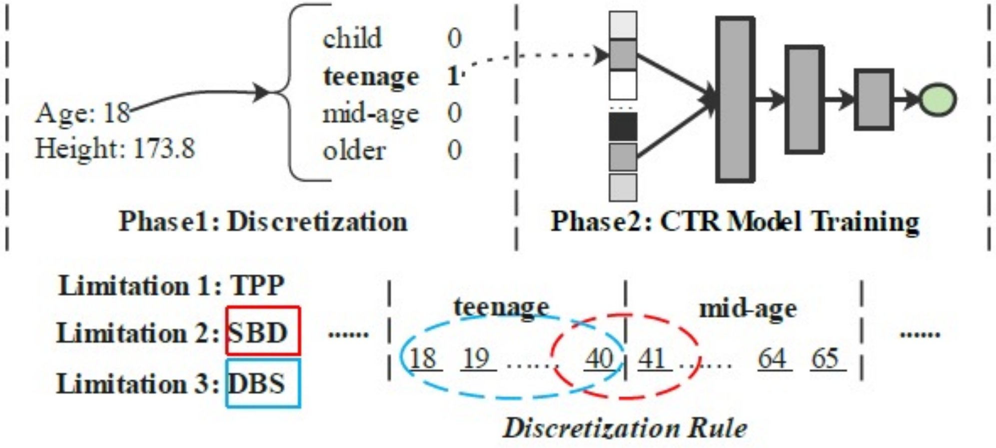
- **Low capacity**: difficult to capture informative knowledge of numerical fields.
- **Poor compatibility**: difficult to adapt to some models (e.g., FM).

2. Category 2: Field Embedding

- **Low capacity**: single shared field-specific embedding.

3. Category 3: Discretization

- **TPP** (Two-Phase Problem)
- **SBD** (Similar value But Dis-similar embedding)
- **DBS** (Dis-similar value But Same embedding)



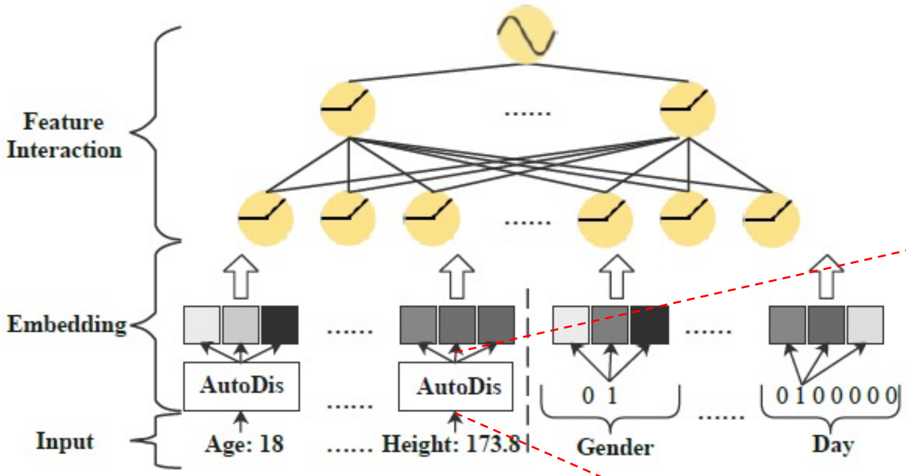
Numerical Embedding-AutoDis

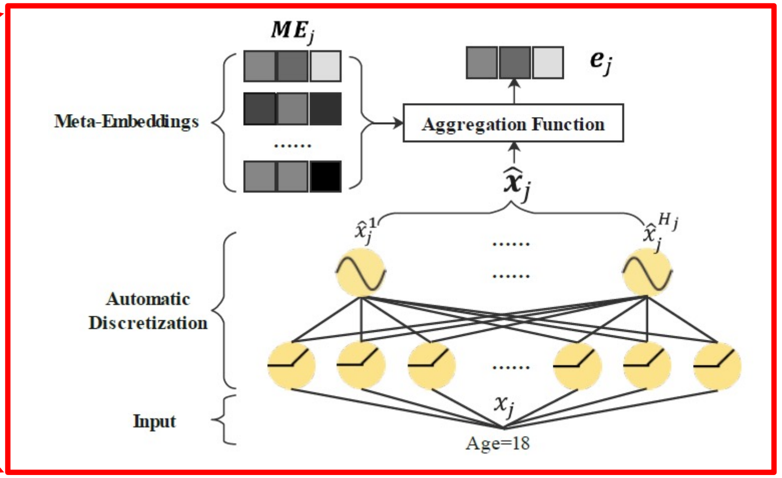
AutoDis is a numerical features embedding learning framework with **high model capacity, end-to-end training and unique representation** properties preserved.

Aggregation Function Meta-embeddings

$$\mathbf{e}_j = \boxed{f\left(\boxed{d_j^{Auto}(x_j)}, \boxed{\mathrm{ME}_j}\right)},$$

Automatic Discretization





Summarize DRS Embedding

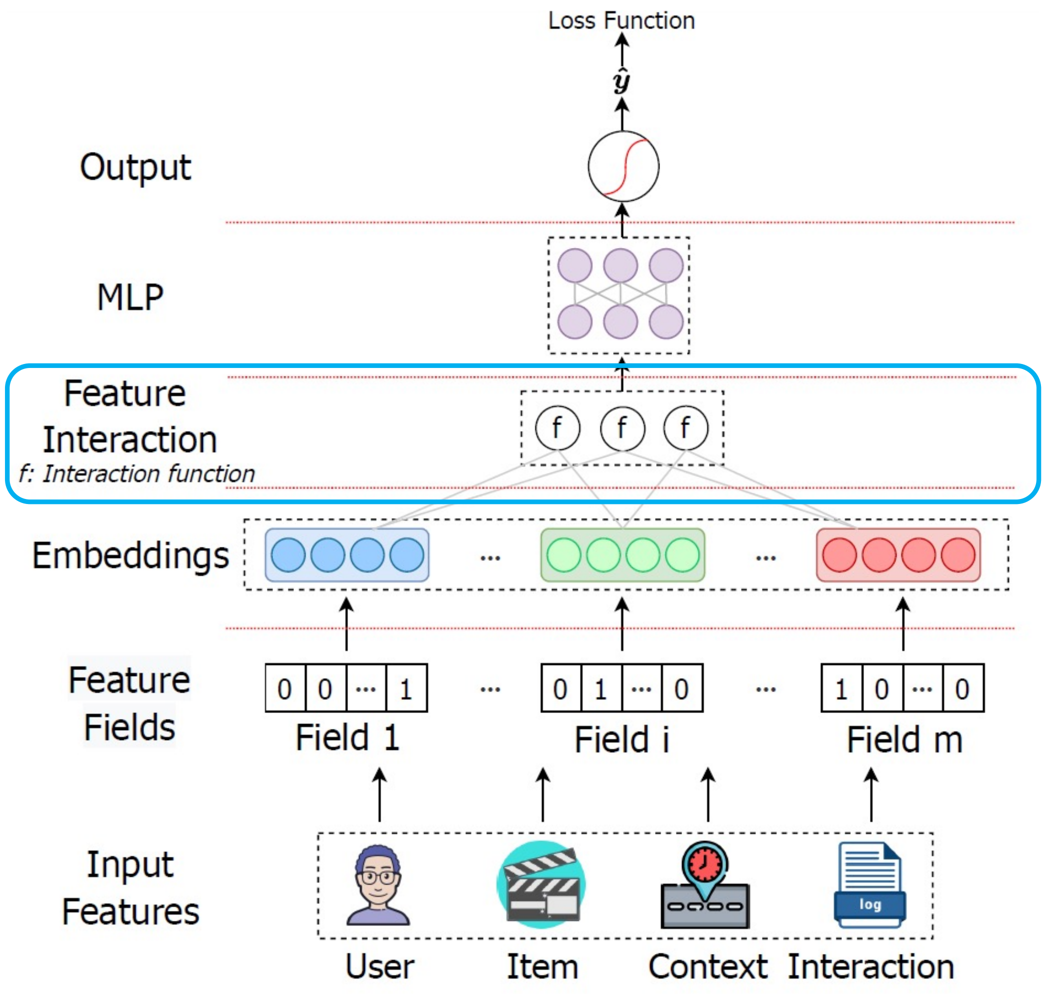
	Model	Feature Field	Search Space	Search Strategy
Single Embedding Search	AMTL	Categorical	d^V	Gradient
	PEP	Categorical	2^{Vd}	Regularization
	AutoEmb	Categorical	a^V	Gradient
	ESPAN	Categorical	a^V	Reinforcement Learning
Group Embedding Search	AutoDim	Categorical	a^{m}	Gradient
	DNIS	Categorical	2^{bd}	Gradient
	NIS	Categorical	b^{a}	Reinforcement Learning
-	AutoDis	Numerical	2^{km}	Gradient

* d is the embedding size, V is the vocabulary size, m is the number of feature fields, a is the number of sub-dimensions, b is the number of groups, k is the number of meta-embeddings. (a < d, b<<V)

Table of Contents

Background

Effectively modelling **feature interactions** is important.



- Both low-order and high-order feature interactions play important roles to model user preference.
  - People like to download popular apps → id of an app may be a signal
  - People often download apps for food delivery at meal time → interaction between app category and time-stamp may be a signal
  - Male teenagers like shooting game → interaction of app category, user gender and age may be a signal
- Most feature interactions are hidden in data and difficult to identify (e.g., “diaper and beer" rule)

Background

The challenges of modelling **feature interactions**:

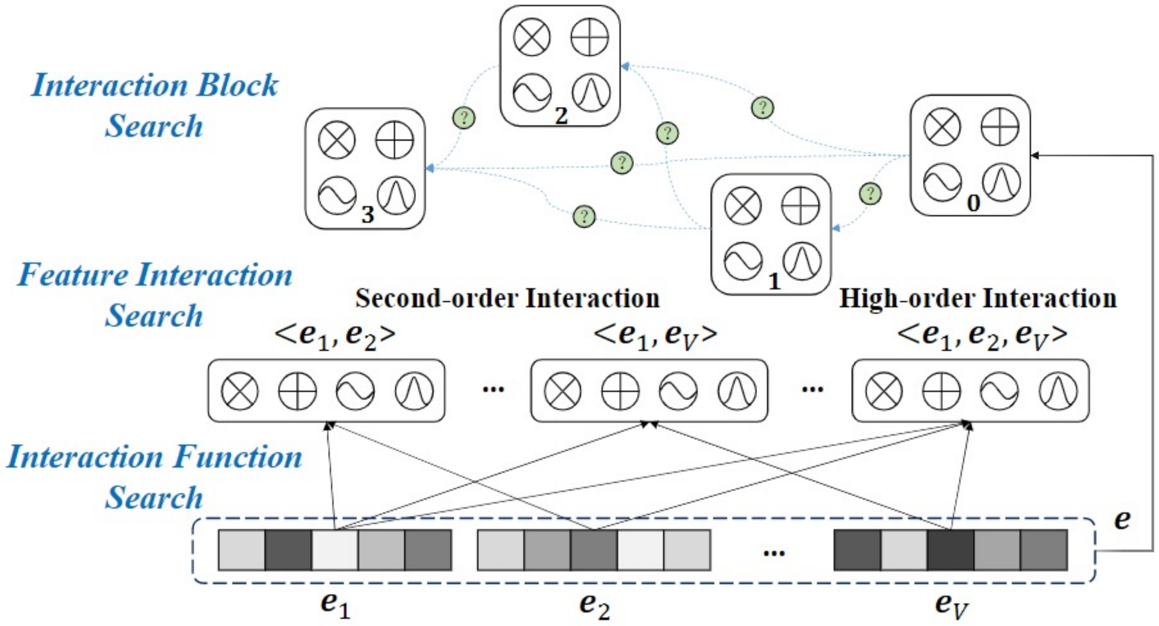
1) Enumerate all feature interactions
    - Large memory and computation cost
    - Difficult to be extended into high-order interactions
    - Useless interactions

2) Require human efforts to identify important **feature interactions**
    - High labor cost
    - Risks missing some counterintuitive (but important) interactions

3) Require human efforts to select appropriate **interaction functions**
    - Human expert knowledge
    - Global interaction function for all the feature interactions

Background

Automatically **select important feature interactions with appropriate interaction functions**

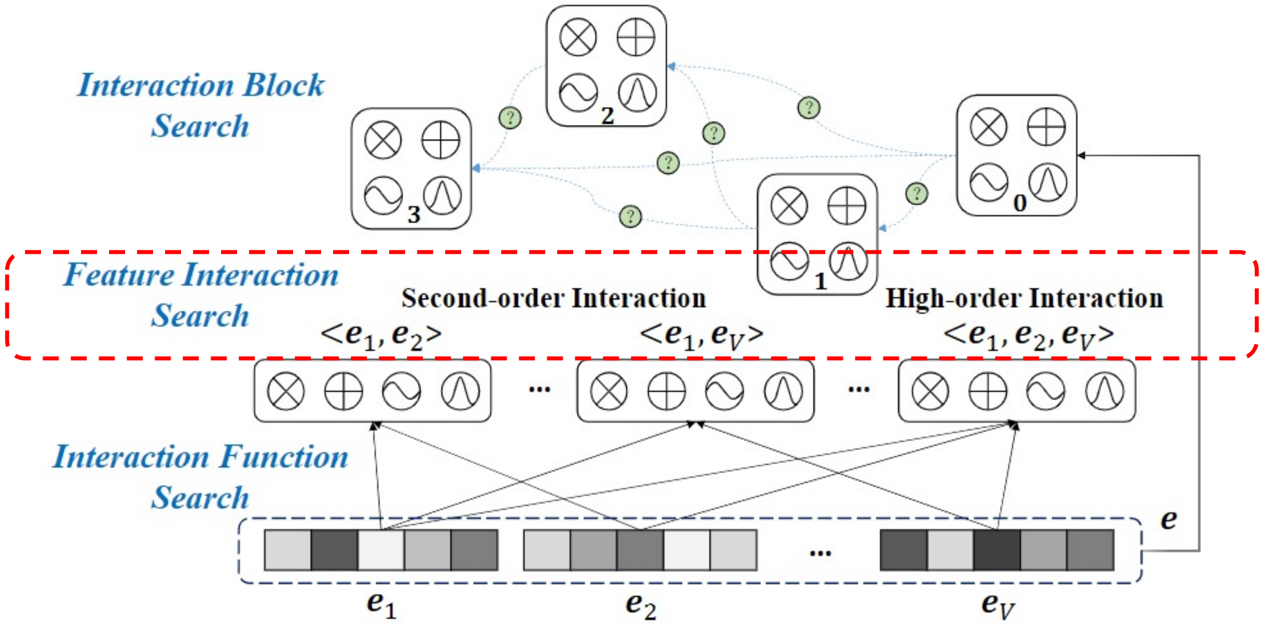


AutoML for feature interaction search:

1. Feature Interaction Search —— search beneficial feature interactions
2. Interaction Function Search —— search suitable interaction functions
3. Interaction Block Search —— search operations over the whole representation

Feature Interaction Search

Automatically **select important feature interactions with appropriate interaction functions**

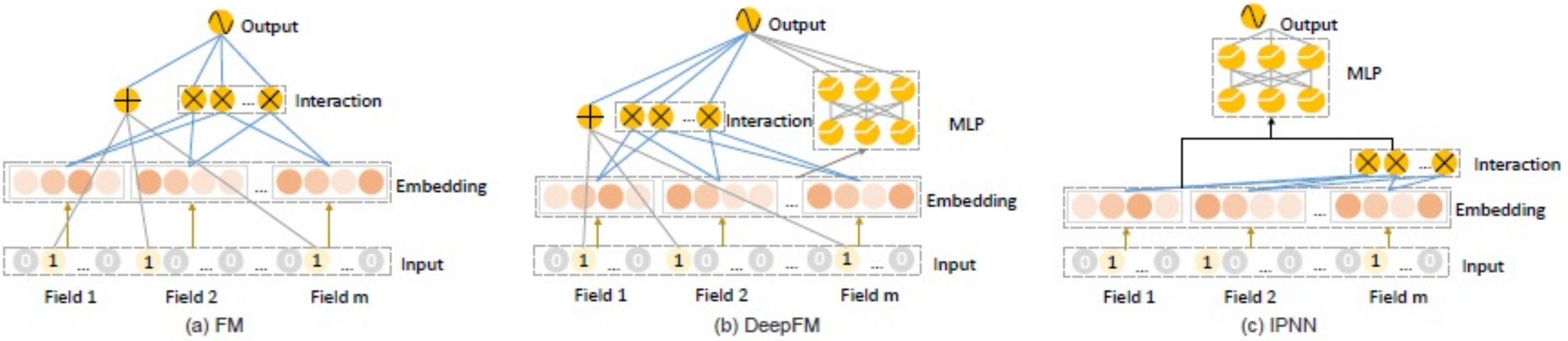


AutoML for feature interaction search:

1. **Feature Interaction Search —— search beneficial feature interactions**
2. Interaction Function Search —— search suitable interaction functions
3. Interaction Block Search —— search operations over the whole representation

Feature Interaction Search-AutoFIS

- Not all the feature interactions are useful.
- Identify such noisy feature interactions and filter them.



AutoFIS: Automatic Feature Interaction Selection in Factorization Models for Click-Through Rate Prediction. KDD 2020

Feature Interaction Search-AutoFIS

- Search Stage
  - Detect useful feature interactions

- Retrain Stage
  - Retrain model with selected feature interactions

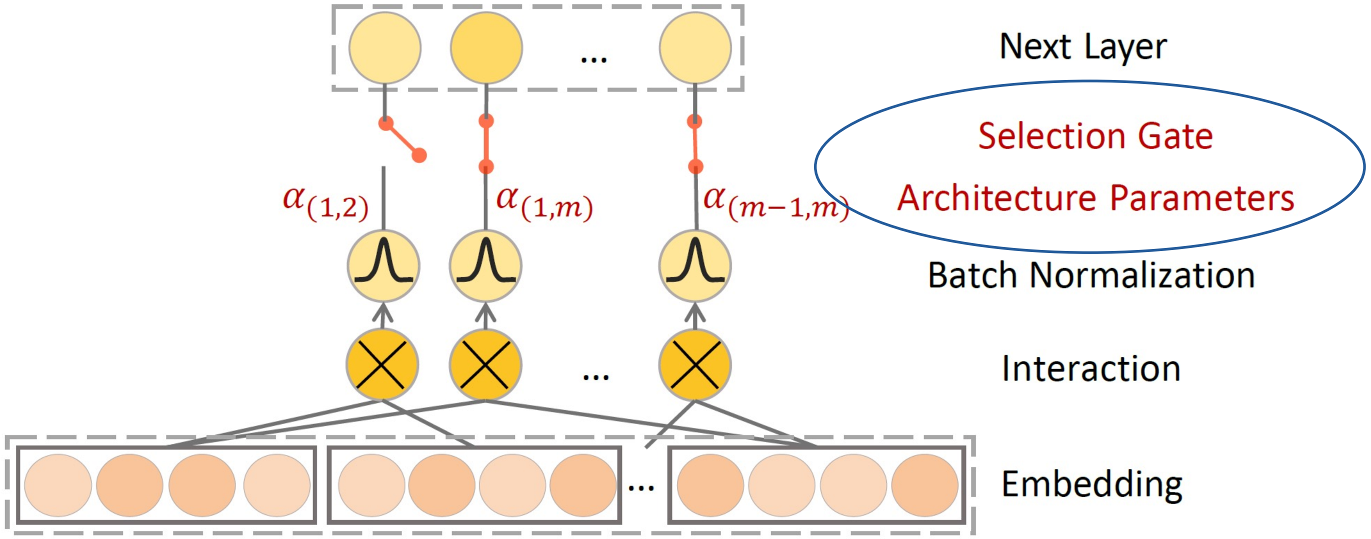
Feature Interaction Search-AutoFIS

Search Stage:

- Gate for each feature interaction
  - Huge search space $2^{C_m^2}$ (m is the number of feature field)
- To make such process differentiable, AutoFIS relaxes the discrete search space to be continuous, by defining architecture parameters α.
  - Batch Normalization to eliminate scale coupling
  - Using GRDA Optimizer to obtain stable and sparse architecture parameters

$$l_{\mathrm{AutoFIS}} = \langle w, x \rangle + \sum_{i=1}^{m} \sum_{j>i}^{m} \alpha_{(i,j)} \langle \boldsymbol{e}_i, \boldsymbol{e}_j \rangle$$

Indicator α = 0 or 1



Retrain Stage:

- Abandon unimportant feature interactions
- Retrain model

Feature Interaction Search-AutoGroup

The limitation of AutoFIS:

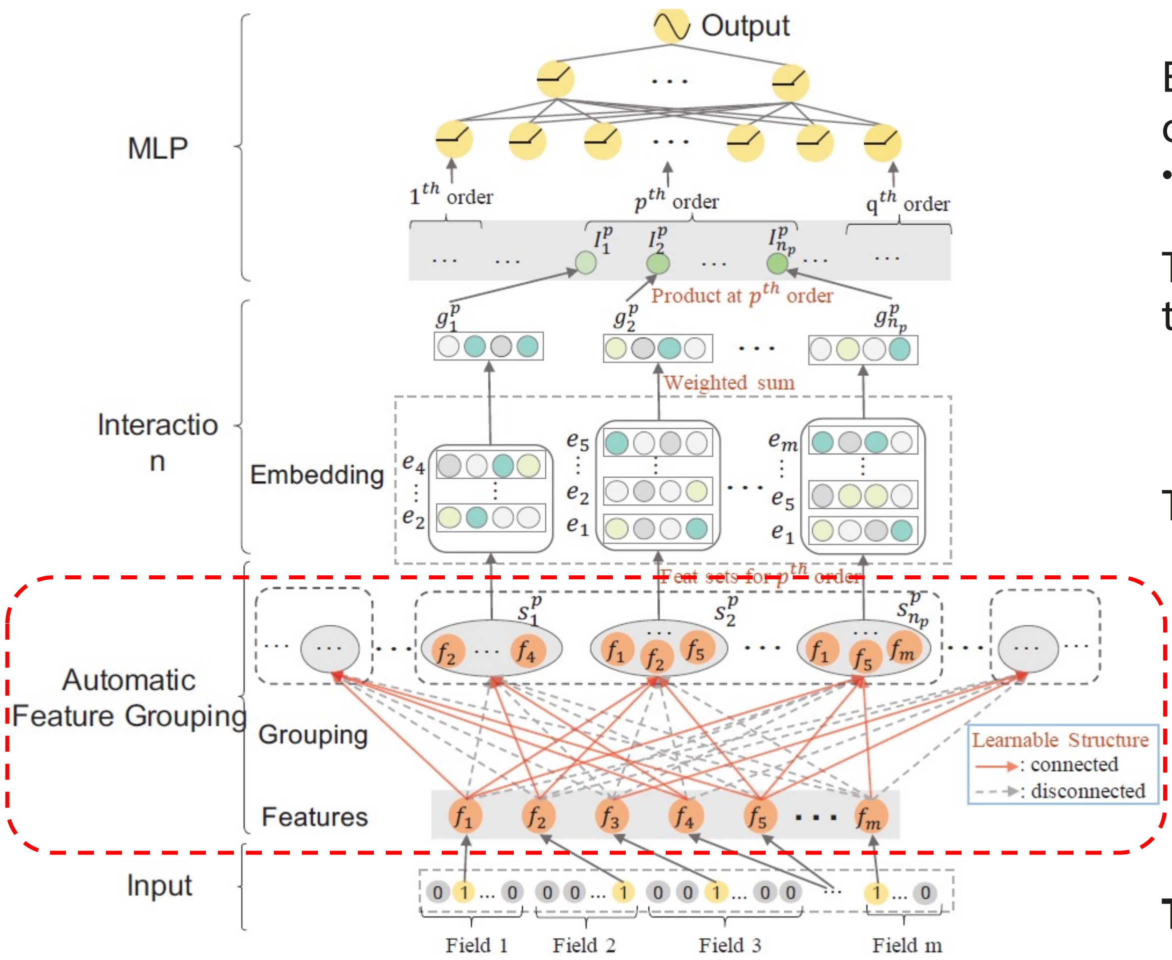
- When searching **high-order feature interactions**, the search space of AutoFIS is huge, resulting in low search efficiency.

Solution of AutoGroup:

- To solve the ***efficiency-accuracy dilemma***, AutoGroup proposes automatic feature grouping, reducing the pth-order search space from $2^{\mathrm{C}_{\mathrm{m}}^{\mathrm{p}}}$ to 2^{gm} (g is the number of pre-defined groups)

Feature Interaction Search-AutoGroup

Feature Grouping Stage:



Each feature is possible to be selected into the feature sets of each order.

- $\Pi_{i,j}^{p} \in \{0,1\}$: whether select feature f_i into the j^{th} set of order-p.

To make the selection differentiable, we relax the binary discrete value to a softmax over the two possibilities:

$$\overline{\Pi}_{i,j}^{p} = \frac{1}{1+\exp(-\alpha_{i,j}^{p})}\Pi_{i,j}^{p} + \frac{\exp\left(-\alpha_{i,j}^{p}\right)}{1+\exp\left(-\alpha_{i,j}^{p}\right)}(1-\Pi_{i,j}^{p}).$$

To learn a less-biased selection probability, we use Gumbel-Softmax:

$$\left(\overline{\Pi}_{i,j}^{p}\right)_{o} = \frac{\exp(\frac{\log\alpha_o + G_o}{\tau})}{\sum_{o'\in\{0,1\}}\exp(\frac{\log\alpha_{o'} + G_{o'}}{\tau})} \text{ where } o \in \{0,1\}.$$

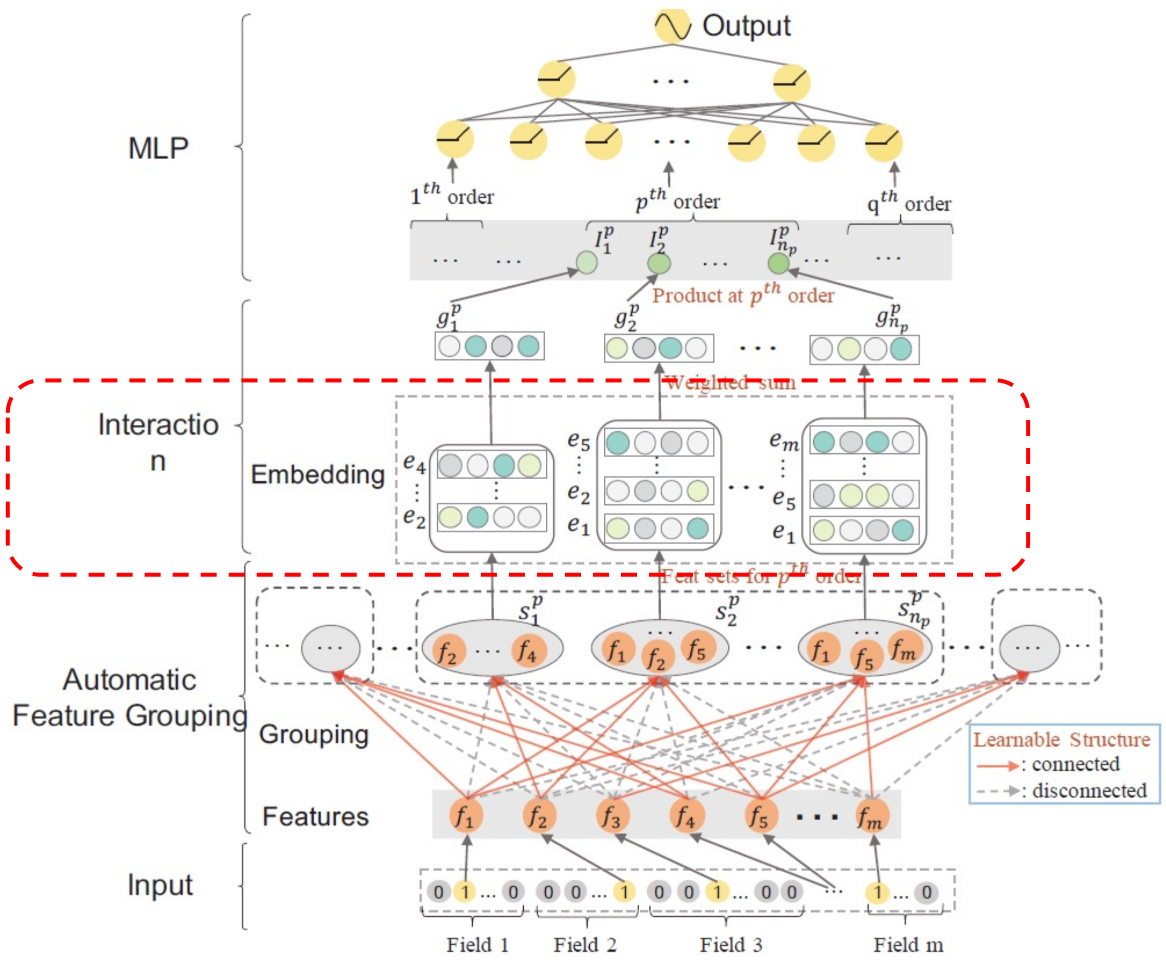
$$\alpha_0 = \frac{1}{1+\exp(-\alpha_{i,j}^{p})} \qquad \alpha_1 = \frac{\exp\left(-\alpha_{i,j}^{p}\right)}{1+\exp\left(-\alpha_{i,j}^{p}\right)}$$

$$G_o = -\log(-\log u) \text{ where } u \sim Uniform(0,1)$$

Trainable Parameters: $\{\alpha_{i,j}^{p}\}$

Feature Interaction Search-AutoGroup

Interaction Stage:



Feature set representation:

$$g_j^p = \sum_{f_i \in s_j^p} w_i^p \, e_i$$

s_j^p: the j^{th} feature set for order-p feature interactions.
e_i: embedding for feature f_i
w_i^p: weights of embeddings in feature set s_j^p.

Interaction at a given order:

- The order-p interaction in a given set s_j^p is:

$$I_j^p = \begin{cases} (g_j^p)^p - \sum_{f_i \in s_j^p} (w_i^p e_i)^p \in R, & p \geq 2 \\ g_j^p \in R^k, & p = 1 \end{cases}$$

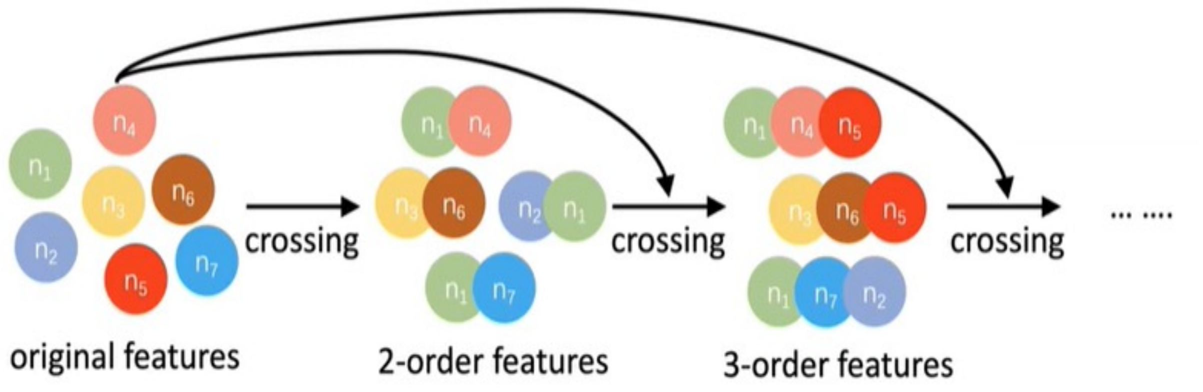
Feature Interaction Search-FIVES

The limitation of AutoGroup:

- Solve the ***efficiency-accuracy dilemma*** via feature grouping
- Ignore the ***Order-priority property***
  - ➢ The higher-order feature interactions quality can be relevant to their de-generated low-order ones

Solution of FIVES:

- Regard the original features as a **feature graph** and model the high-order feature interactions by the multilayer convolution of **GNN**, reducing the pth-order search space from $2^{C_m^p}$ to 2^{m^2}.
- Parameterize the **adjacency matrix** and make them depend on the previous layer.



FIVES: Feature Interaction via Edge Search for Large-Scale Tabular Data. KDD 2021

Feature Interaction Search-FIVES

- With an adjacency tensor A, the dedicated **graph convolutional** operator produces the node representations **layer-by-layer**. For the k-order:

$$\mathbf{n}_i^{(k)} = \mathbf{p}_i^{(k)} \odot \mathbf{n}_i^{(k-1)}$$
$$\text{where} \quad \mathbf{p}_i^{(k)} = \text{MEAN}_{j|\mathrm{A}_{i,j}^{(k)}=1}\{\mathbf{W}_j \mathbf{n}_j^{(0)}\}.$$

- The node representation at k-th layer corresponds to the generated $(k+1)$-order interactive features:

$$\mathbf{n}_i^{(k)} = \text{MEAN}_{j|\mathrm{A}_{i,j}^{(k)}=1}\{\mathbf{W}_j \mathbf{n}_j^{(0)}\} \odot \mathbf{n}_i^{(k-1)}$$
$$\approx \text{MEAN}_{(c_1,\ldots,c_k)|\mathrm{A}_{i,c_j}^{(j)}=1, j=1,\ldots,k}\{f_{c_1} \otimes \cdots \otimes f_{c_k} \otimes f_i\},$$

Feature Interaction Search-FIVES

- The task of generating useful interactive features is equivalent to learning an optimal adjacency tensor A, namely **edge search**.
- The edge search task could be formulated as a **bi-level optimization problem**:

$$\min_{A} \mathcal{L}(\mathcal{D}_{\text{val}}|A,\Theta(A))$$
$$\text{s.t.} \quad \Theta(A) = \arg\min_{\Theta} \mathcal{L}\left(\mathcal{D}_{\text{train}}|A,\Theta\right) \tag{3}$$

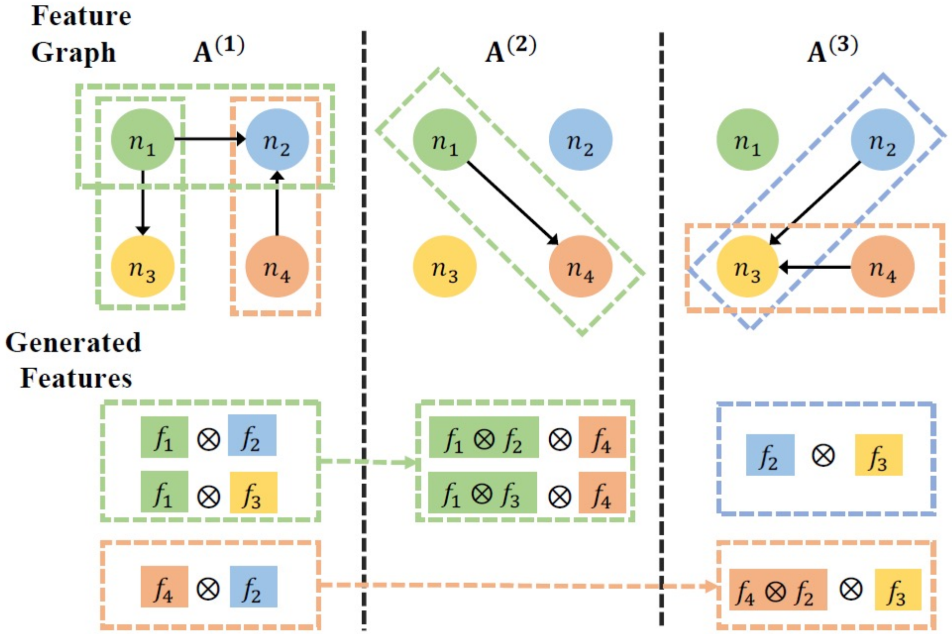
- To make the optimization more efficient, FIVES uses a soft $A^{(k)}$ for propagation at the k-th layer, while the calculation of $A^{(k)}$still depends on a binarized $A^{(k-1)}$:

$$A^{(k)} \triangleq \left(D^{(k-1)}\right)^{-1} \varphi\left(A^{(k-1)}\right) \sigma\left(H^{(k)}\right) \tag{4}$$

Degree matrix of $A^{(k-1)}$

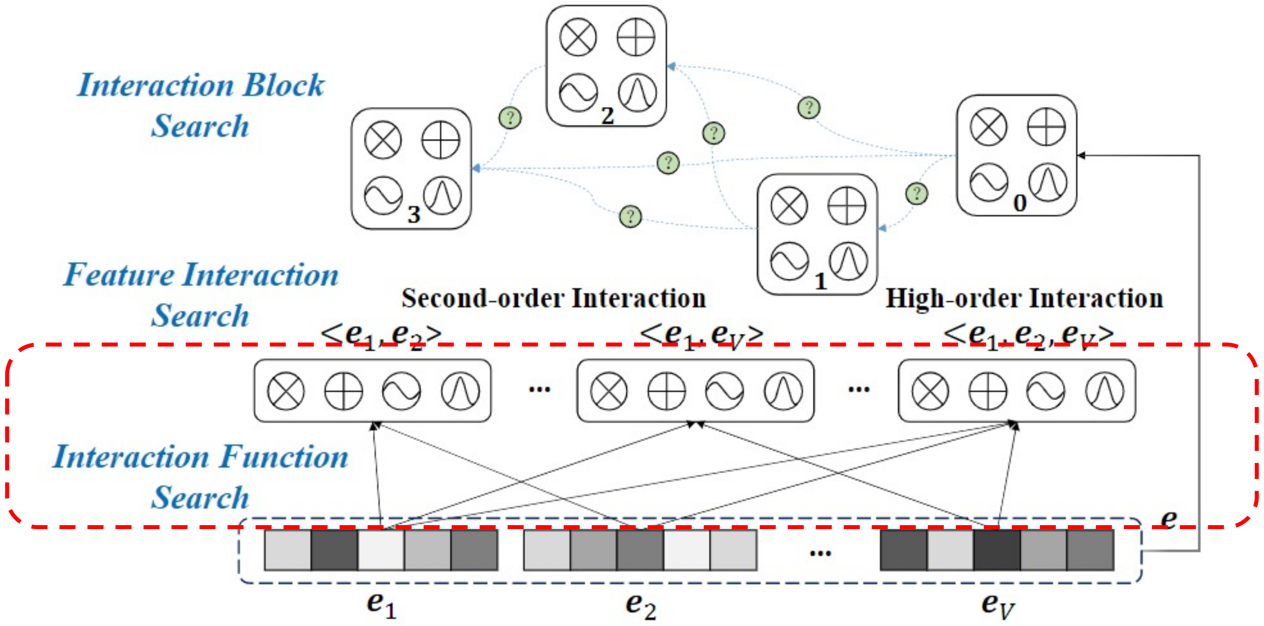
Binarize the soft $A^{(k-1)}$

Interactions at k-th layer



Interaction Function Search

Automatically **select important feature interactions with appropriate interaction functions**



AutoML for feature interaction search:

1. Feature Interaction Search —— search beneficial feature interactions
2. **Interaction Function Search —— search suitable interaction functions**
3. Interaction Block Search —— search operations over the whole representation

Interaction Function Search-SIF

- Generate embedding vectors for users and items
- Generate predictions by an **inner product** between embedding vectors
- Evaluate predictions by a loss function on the training data set

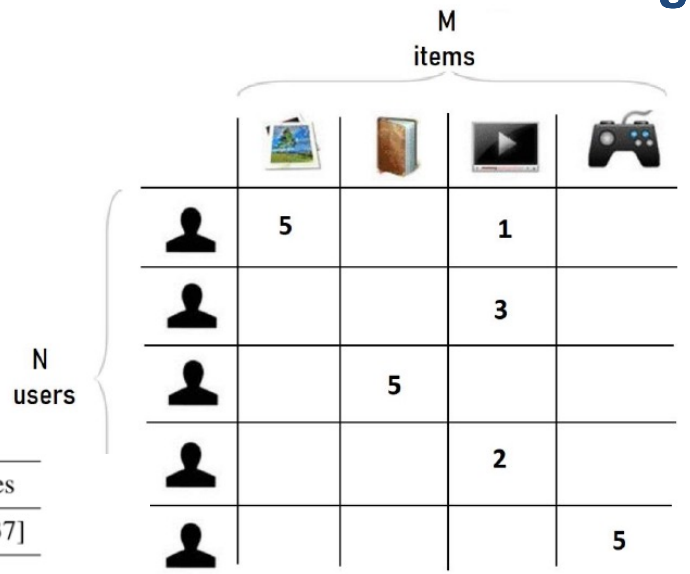
Interaction function:
How embedding vectors interact with each other?

	IFC	operation	space	predict time	recent examples
human-designed	$\langle \boldsymbol{u}_i, \boldsymbol{v}_j \rangle$	inner product	$O((m+n)k)$	$O(k)$	MF [28], FM [37]
	$\boldsymbol{u}_i - \boldsymbol{v}_j$	plus (minus)	$O((m+n)k)$	$O(k)$	CML [19]
	$\max(\boldsymbol{u}_i, \boldsymbol{v}_j)$	max, min	$O((m+n)k)$	$O(k)$	ConvMF [25]
	$\sigma([\boldsymbol{u}_i; \boldsymbol{v}_j])$	concat	$O((m+n)k)$	$O(k)$	Deep&Wide [9]
	$\sigma(\boldsymbol{u}_i \odot \boldsymbol{v}_j + \boldsymbol{H}[\boldsymbol{u}_i; \boldsymbol{v}_j])$	multi, concat	$O((m+n)k)$	$O(k^2)$	NCF [17]
	$\boldsymbol{u}_i * \boldsymbol{v}_j$	conv	$O((m+n)k)$	$O(k\log(k))$	ConvMF [25]
	$\boldsymbol{u}_i \otimes \boldsymbol{v}_j$	outer product	$O((m+n)k)$	$O(k^2)$	ConvNCF [16]
AutoML	SIF (proposed)	searched	$O((m+n)k)$	$O(k)$	——

Is there an absolute best IFC? : NO, depends on tasks and datasets [1]

Efficient Neural Interaction Function Search for Collaborative Filtering. WWW 2020

Collaborative filtering

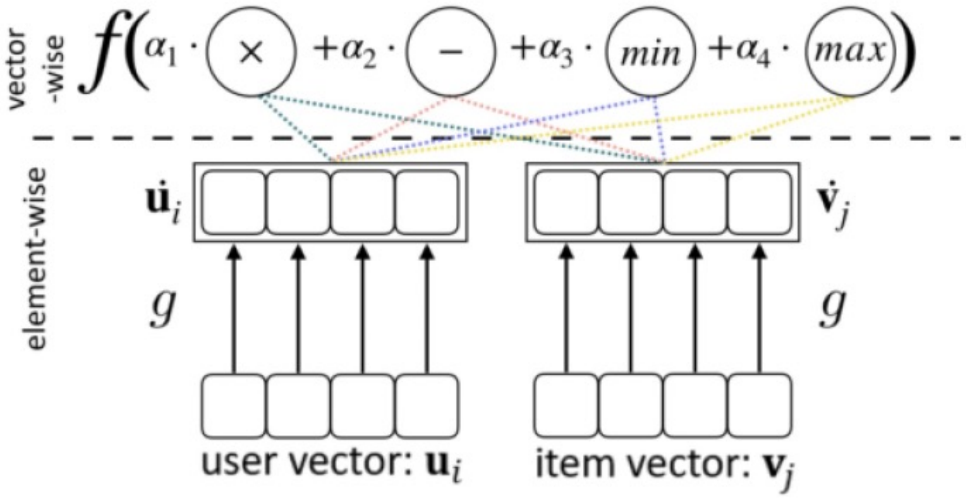


Interaction Function Search-SIF

- SIF selects different interaction functions across different datasets.

IFC	operation
$\langle \boldsymbol{u}_i, \boldsymbol{v}_j \rangle$	inner product
$\boldsymbol{u}_i - \boldsymbol{v}_j$	plus (minus)
$\max(\boldsymbol{u}_i, \boldsymbol{v}_j)$	max, min
$\sigma([\boldsymbol{u}_i; \boldsymbol{v}_j])$	concat
$\sigma(\boldsymbol{u}_i \odot \boldsymbol{v}_j + \boldsymbol{H}[\boldsymbol{u}_i; \boldsymbol{v}_j])$	multi, concat
$\boldsymbol{u}_i * \boldsymbol{v}_j$	conv
$\boldsymbol{u}_i \otimes \boldsymbol{v}_j$	outer product

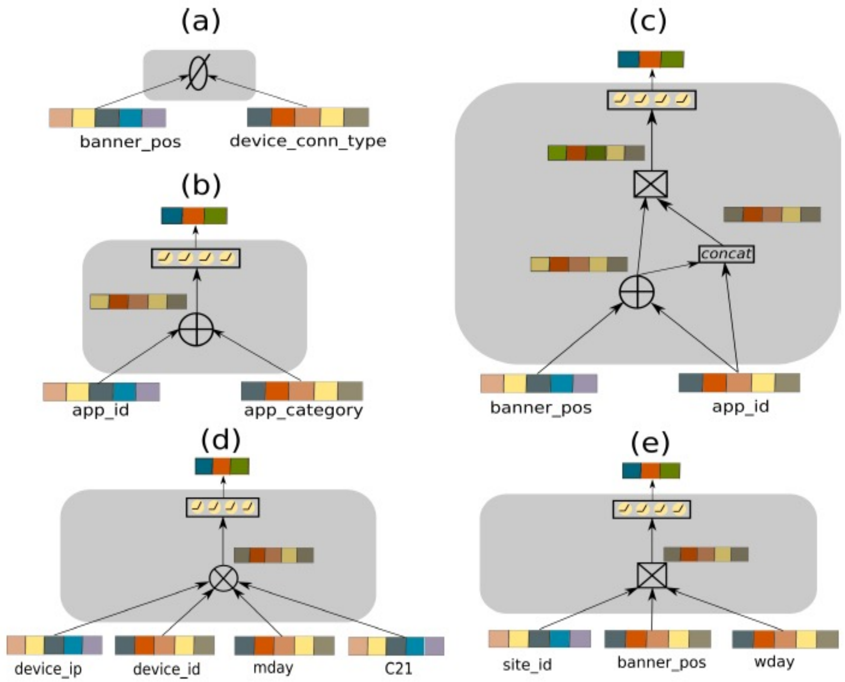
Cut the search space into two blocks

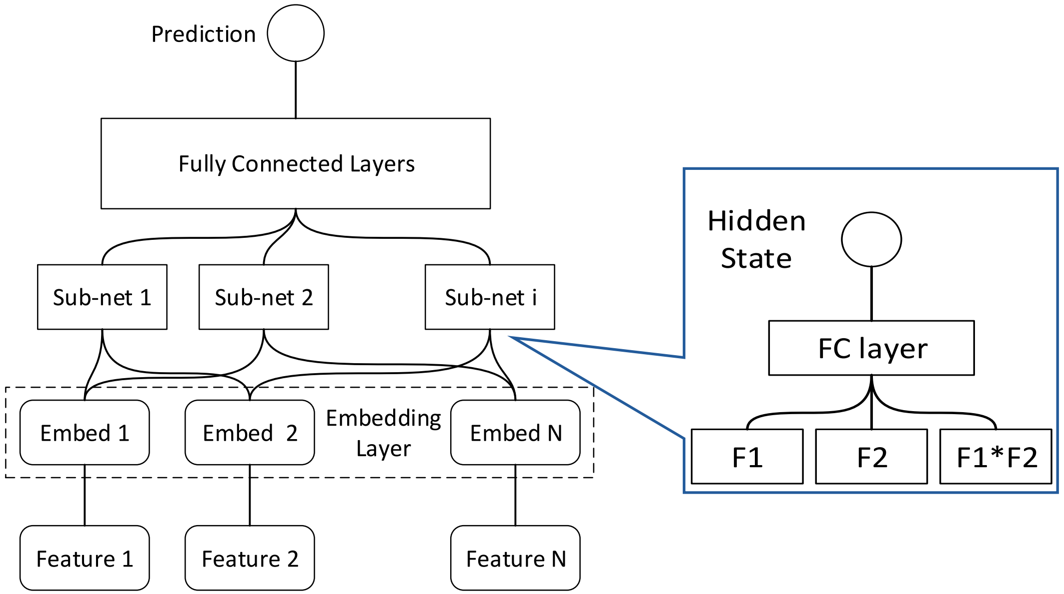


- Vector-level: simple linear algebra operations
- Elementwise: shared nonlinear transformation

Interaction Function Search-AutoFeature

- Not all the feature interactions between each pair of fields need to be modeled.
- Not all the useful feature interactions can be modeled by the same interaction functions.



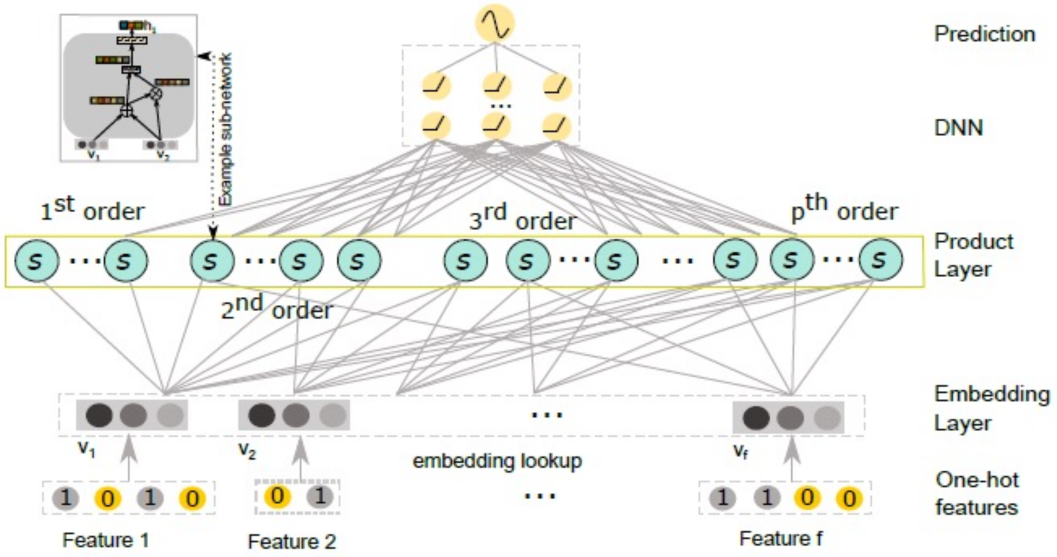


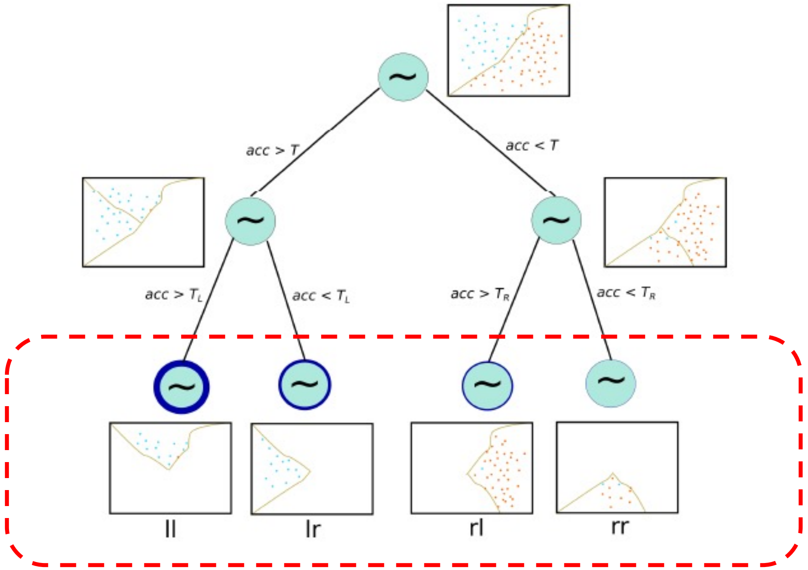
AutoFeature: Searching for Feature Interactions and Their Architectures for Click-through Rate Prediction. CIKM 2020

Interaction Function Search-AutoFeature

AutoFeature automatically designs a different sub-net for each pair of fields.

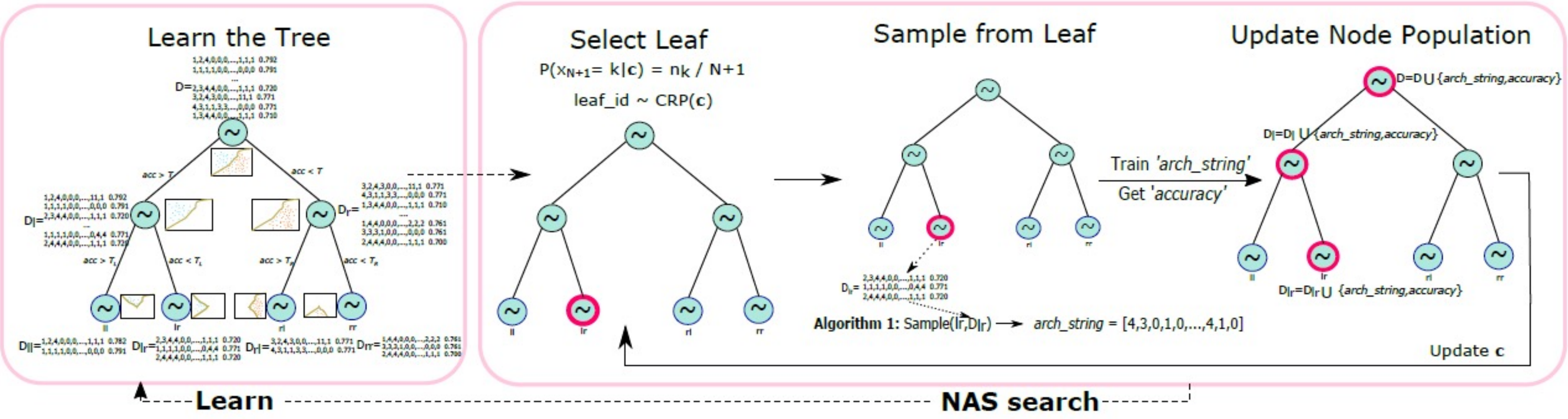
- **Train a Naïve Bayes Tree** to classify different network structures, where the tree tends to classify the **most well-performed network** into the **leftmost leaf subspace**, such that the next generation can be more effective.
- **Sample leaf nodes** from these leaf subspaces based on the Chinese Restaurant Process (CRP).





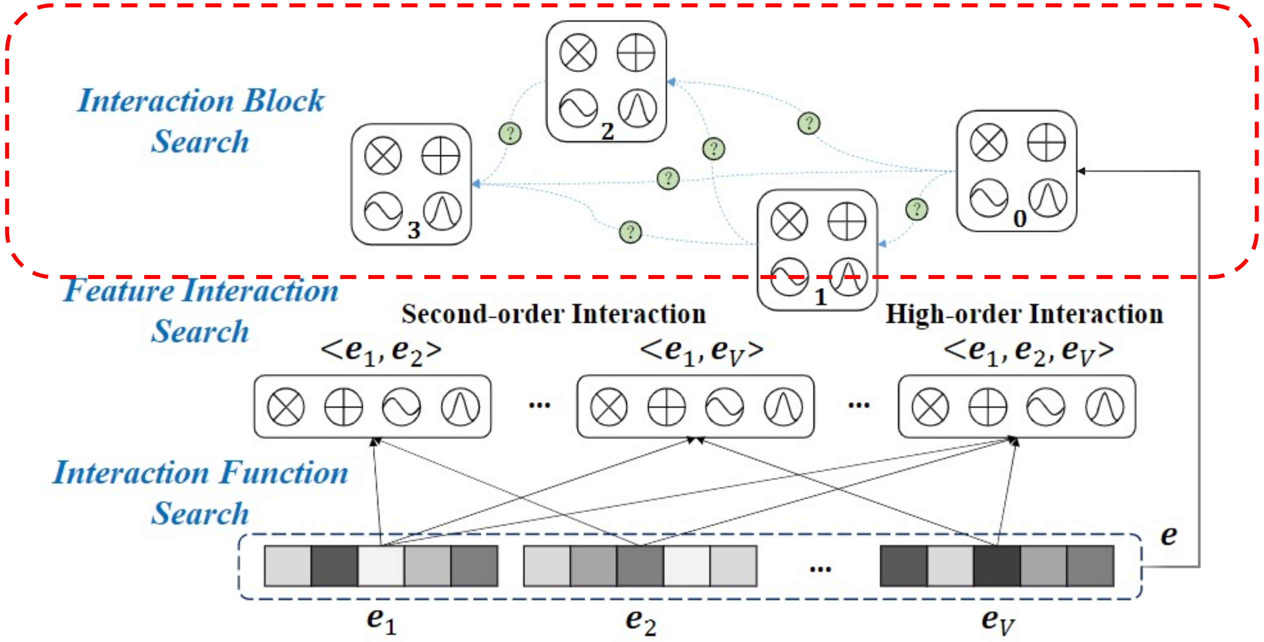
Interaction Function Search-AutoFeature

- The top two samples with the highest accuracy will be picked to perform a **crossover** operation at the midpoint of the architecture string, which is followed by q **mutations**.
- **Check** if the resulting architecture belongs to the subspace represented by the leaf node. If this is not the case then the procedure is repeated.
- The whole search procedure continues until the desired accuracy is achieved or the maximum number of steps is reached.



Interaction Block Search

Automatically **select important feature interactions with appropriate interaction functions**



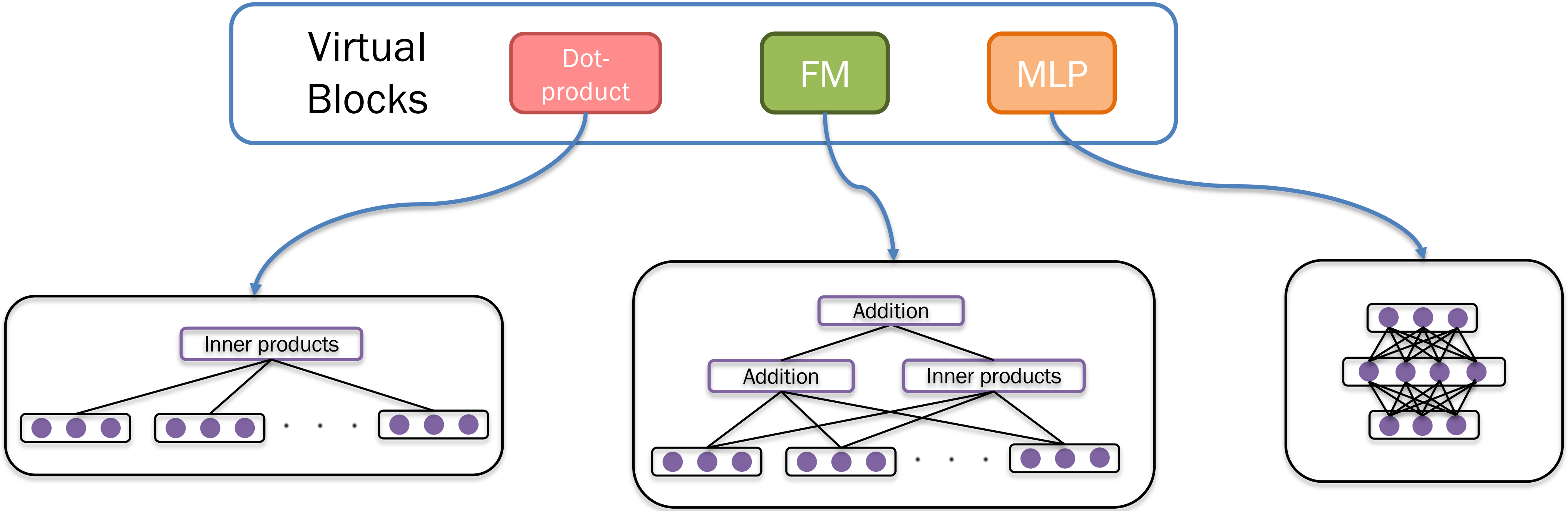
AutoML for feature interaction search:

1. Feature Interaction Search —— search beneficial feature interactions
2. Interaction Function Search —— search suitable interaction functions
3. **Interaction Block Search —— search operations over the whole representation**

Interaction Block Search-AutoCTR

Hierarchical Search Space

- Properties: functionality complementary, complexity aware, …
- Examples: MLP block, dot-product block, factorization-machine block, …

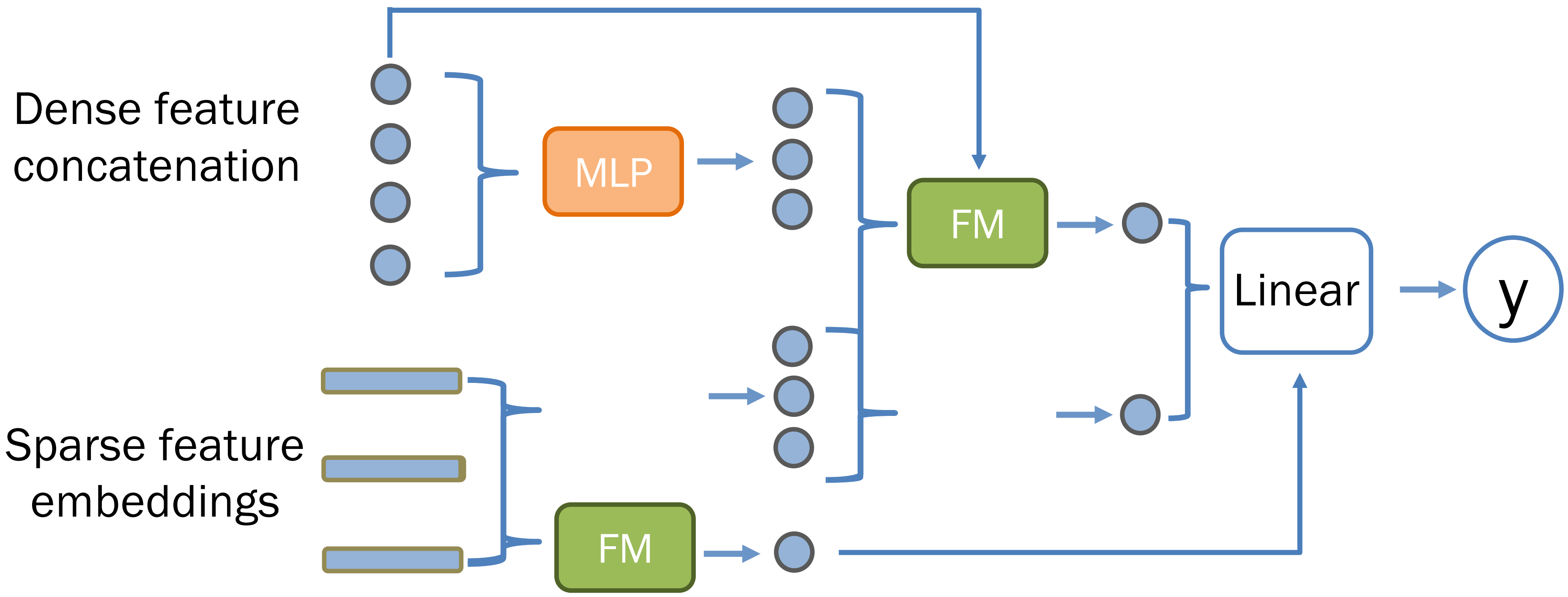


Towards Automated Neural Interaction Discovery for Click-Through Rate Prediction. KDD 2020

Interaction Block Search-AutoCTR

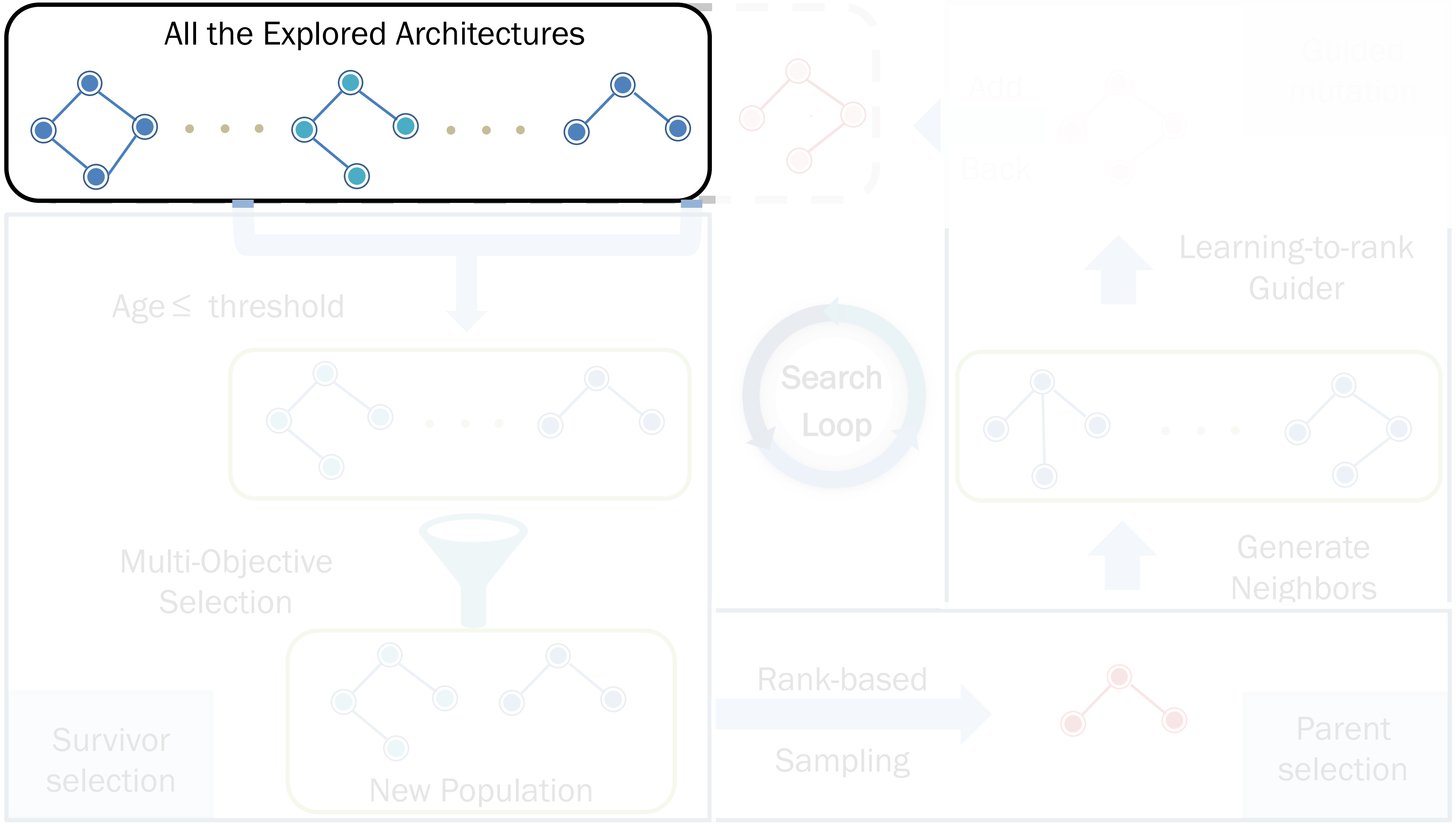
Search space construction

- DAG of virtual blocks and grouped feature embeddings
- Both block hyper-parameters and connection among blocks are to be searched



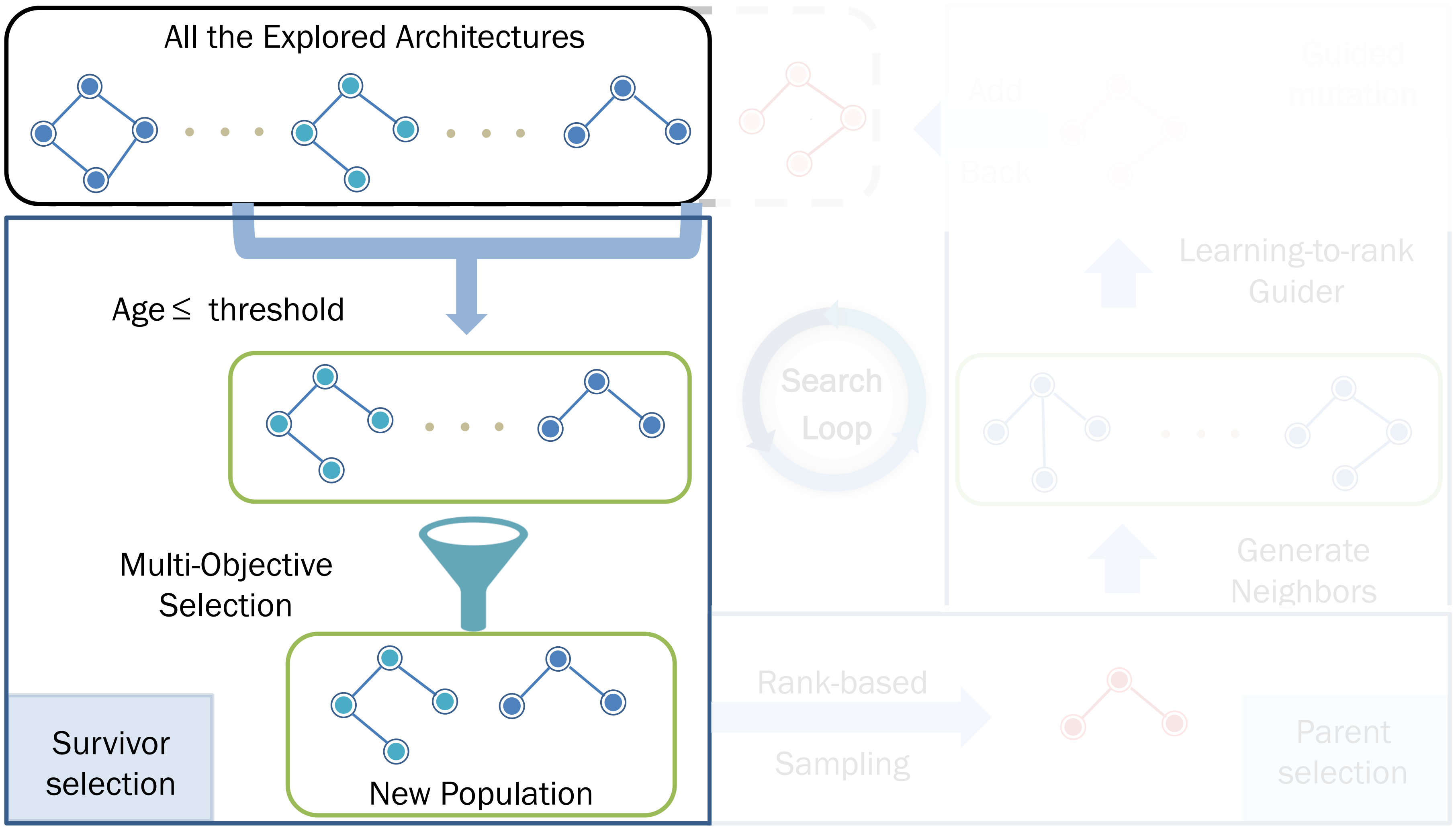
Interaction Block Search-AutoCTR

Multi-Objective Evolutionary Search Algorithm



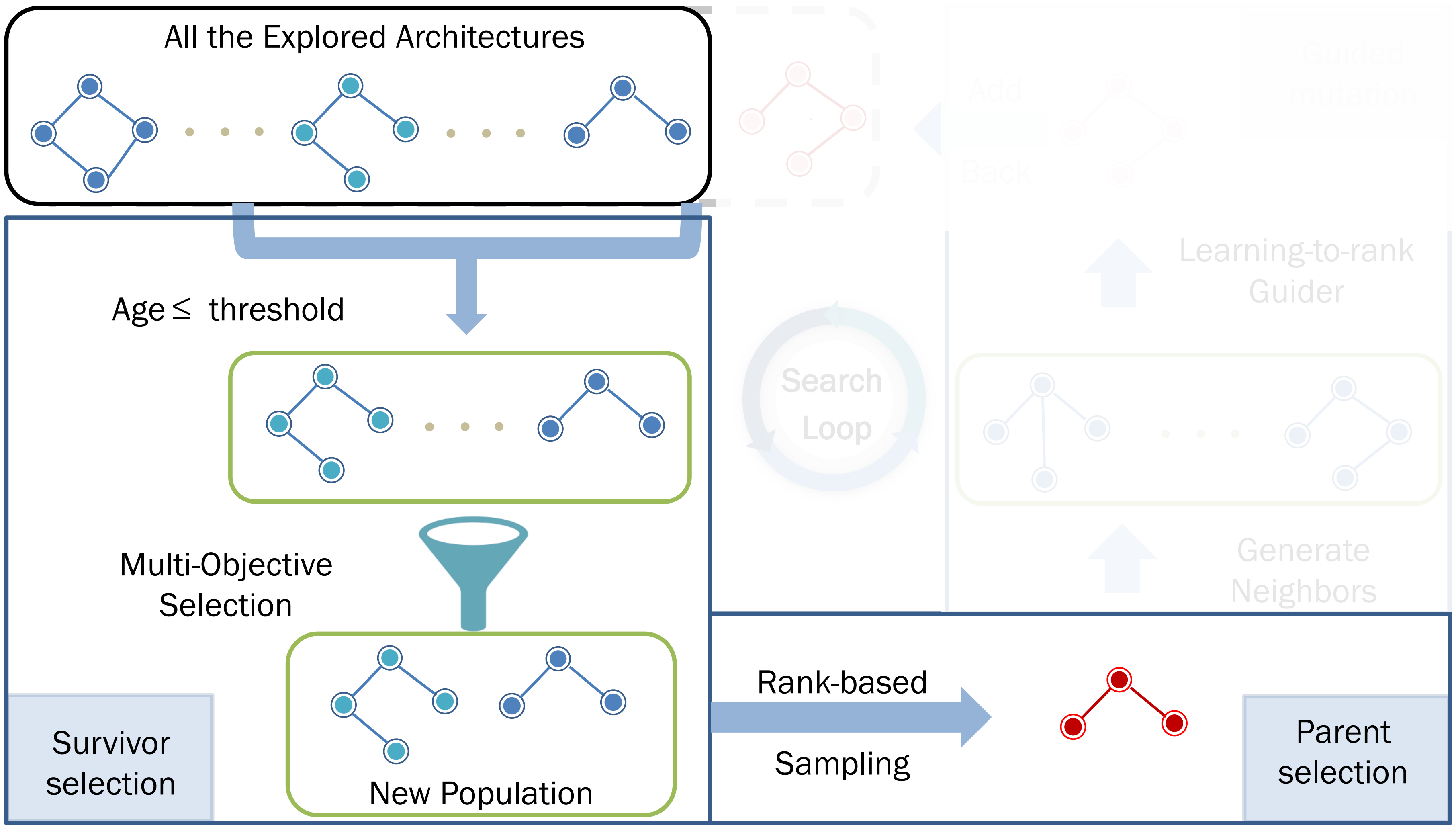
Interaction Block Search-AutoCTR

Multi-Objective Evolutionary Search Algorithm



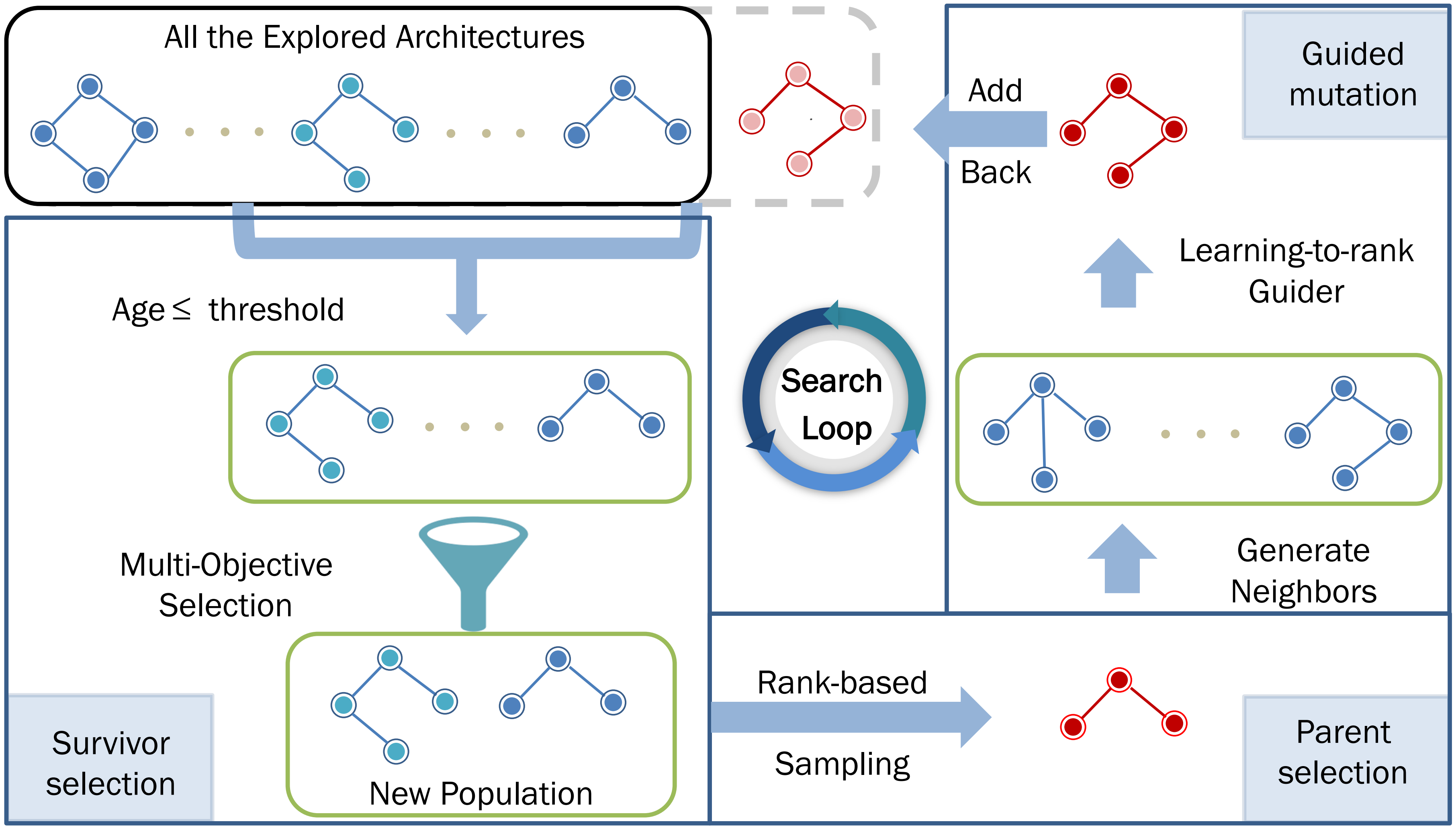
Interaction Block Search-AutoCTR

Multi-Objective Evolutionary Search Algorithm



Interaction Block Search-AutoCTR

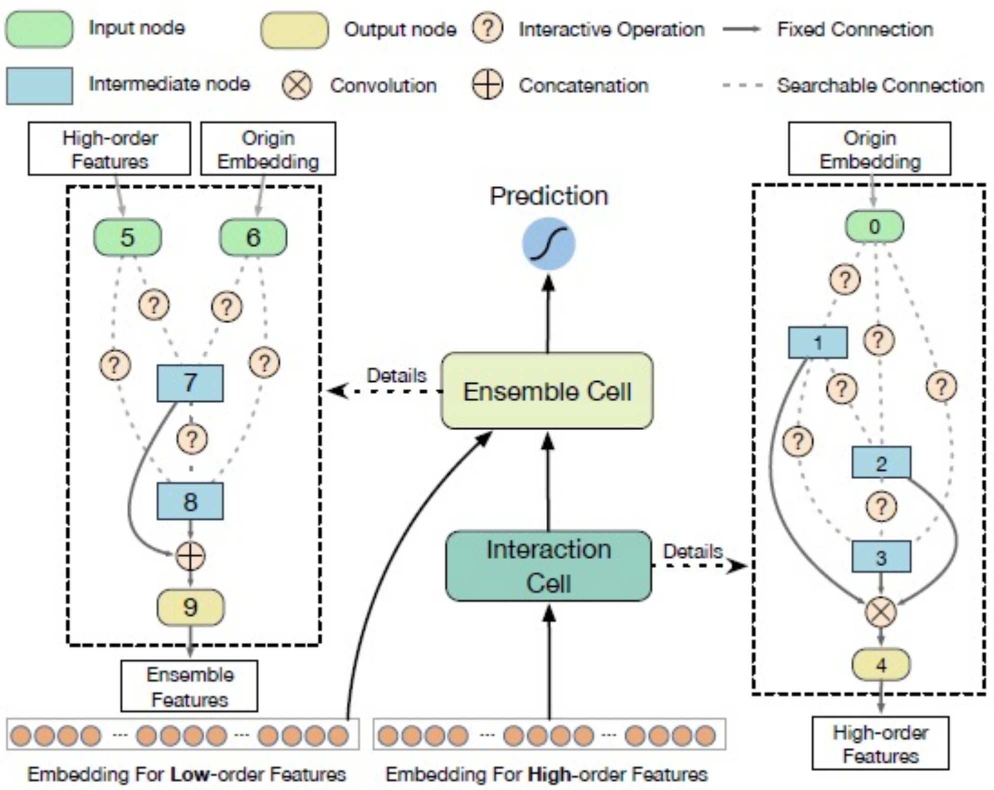
Multi-Objective Evolutionary Search Algorithm



Interaction Block Search-AutoPI

Search Space

- The interaction cell formulates the higher-order feature interactions
- The ensemble cell formulates the ensemble of lower-order and higher-order interactions



- Skip-connection[1]

$$\boldsymbol{E}' = o_{\text{skip}}(\boldsymbol{E}) = \boldsymbol{E} \in \mathbb{R}^{m\times k}$$

- SENET Layer[2]

$$\boldsymbol{E}' = [a_1' \cdot \boldsymbol{x}_1, a_2' \cdot \boldsymbol{x}_2, \cdots, a_m' \cdot \boldsymbol{x}_m] \in \mathbb{R}^{m\times k}$$

- Self-attention[3]

$$\boldsymbol{E}' = [\boldsymbol{e}_1^{\text{Res}}, \boldsymbol{e}_2^{\text{Res}}, \cdots, \boldsymbol{e}_m^{\text{Res}}] \in \mathbb{R}^{m\times k}$$

- FM Layer

$$\boldsymbol{p} = \{(\boldsymbol{x}_j \cdot \boldsymbol{x}_j)\}_{(i,j)\in R_x}$$
$$\text{s.t. } \boldsymbol{R}_x = \{(i,j)\}_{i\in\{1,\cdots,m\}, j\in\{1,\cdots,m\}, i<j}$$

- FC Layer

$$\boldsymbol{E}' = \boldsymbol{e} \cdot \boldsymbol{W}_{\text{FC}}$$

- 1D Conv

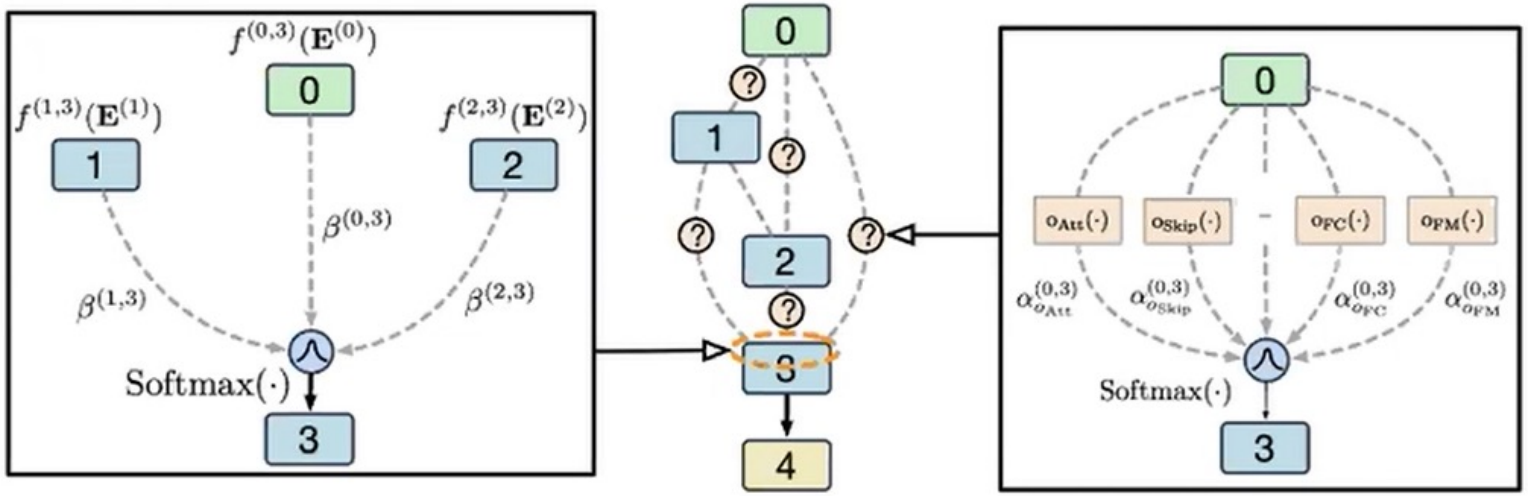
$$\left\{\boldsymbol{C}^{(i)} \in \mathbb{R}^{1\times 1\times m}\right\}_{i\in\{1,\cdots,m\}}$$

A General Method For Automatic Discovery of Powerful Interactions In Click-Through Rate Prediction. SIGIR 2021

Interaction Block Search-AutoPI

Search Strategy

- Continuous relaxation



Continuous relaxation visualization

By introducing the **operator-level** and **edge-level architecture parameters** for continuous relaxation, a **differentiable objective function** will be obtained:

$$\boxed{\begin{aligned} &\min_{\alpha,\beta}\ \mathcal{L}_{val}\left(w^*(\alpha,\beta),\alpha,\beta\right) \\ &\text{s.t. } w^*(\alpha,\beta) = \text{argmin}_w\, \mathcal{L}_{\text{train}}(w,\alpha,\beta) \end{aligned}}$$

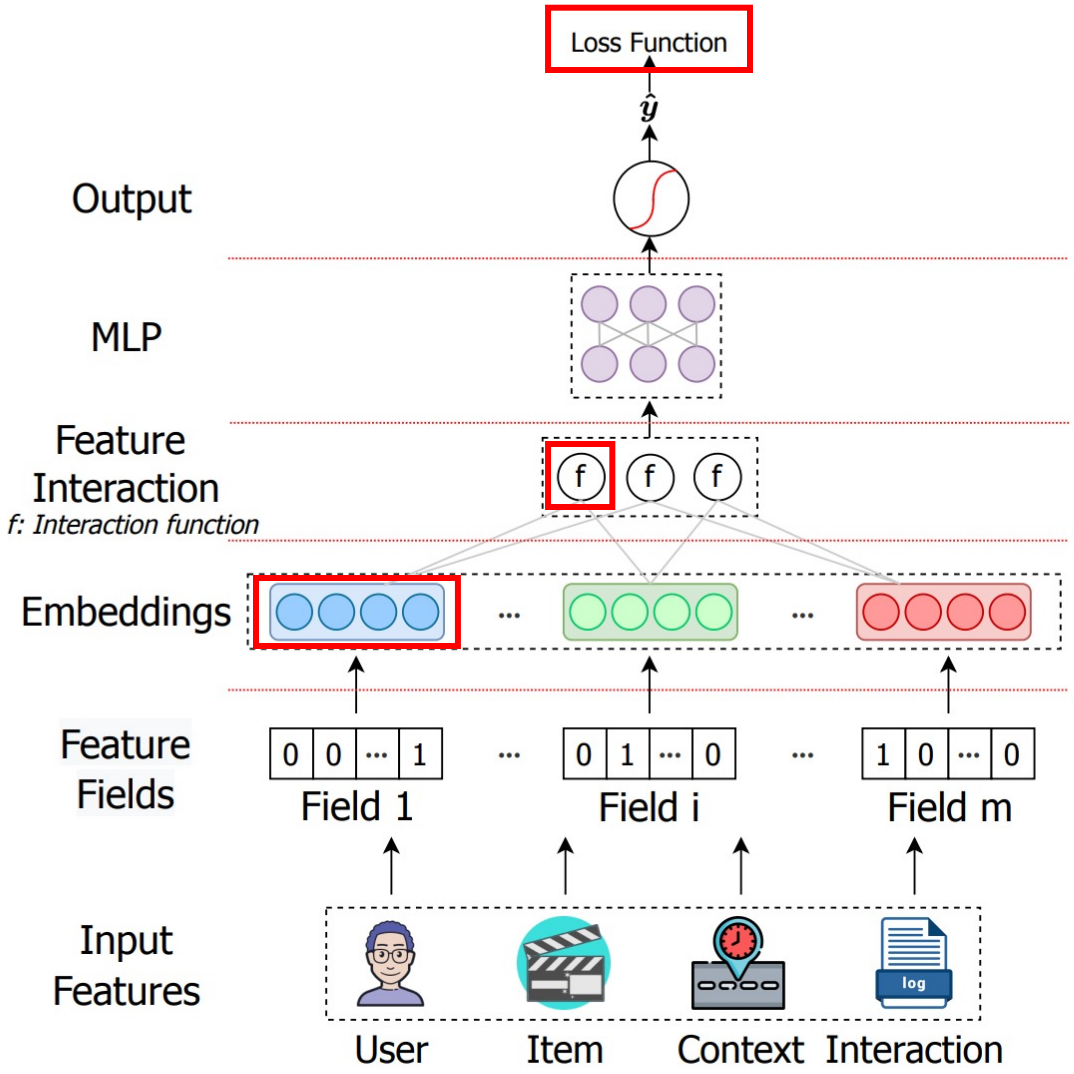
Summarize DRS Interaction

Type	Model	Search Space	Search Strategy
Feature Interaction Search	AutoFIS	$2^{C_m^p}$	Gradient
	AutoGroup	2^{gm}	Gradient
	FIVES	2^{m^2}	Gradient
Interaction Function Search	SIF	b^{a}	Gradient
	AutoFeature	$\mathrm{b}^{C_m^p}$	Evolutionary
Interaction Block Search	AutoCTR	b^{a}	Evolutionary
	AutoPI	b^{a}	Gradient

* m is the number of feature fields, p is the order, g is the number of pre-defined groups, a is the number of pre-defined bolcks, b is the number of candidate interaction functions.

Table of Contents

Background



- Comprehensive search:
  Searching for several parts of DRS
- System design:
  Searching for architectures other than aforementioned parts

Comprehensive Search & System

Type	Model	Search Space	Search Strategy
Comprehensive Search	AMEIR	Sequential model, Feature interaction, MLP	One-shot Random Search
	AIM	Embedding Dimension, Intearaction Function, Feature Interaction	Gradient
	AutoIAS	Embedding Dimension, Projection Dimension, Interaction Function, Feature Interaction, MLP	Reinforcement Learning
System Design	AutoLoss	Optimization: Loss Function	Gradient
	AutoGSR	Structure Design: GNN Architecture	Gradient
	AutoFT	Parameter Tuning: Fine-Tune or Not (For pre-trained models)	Gradient

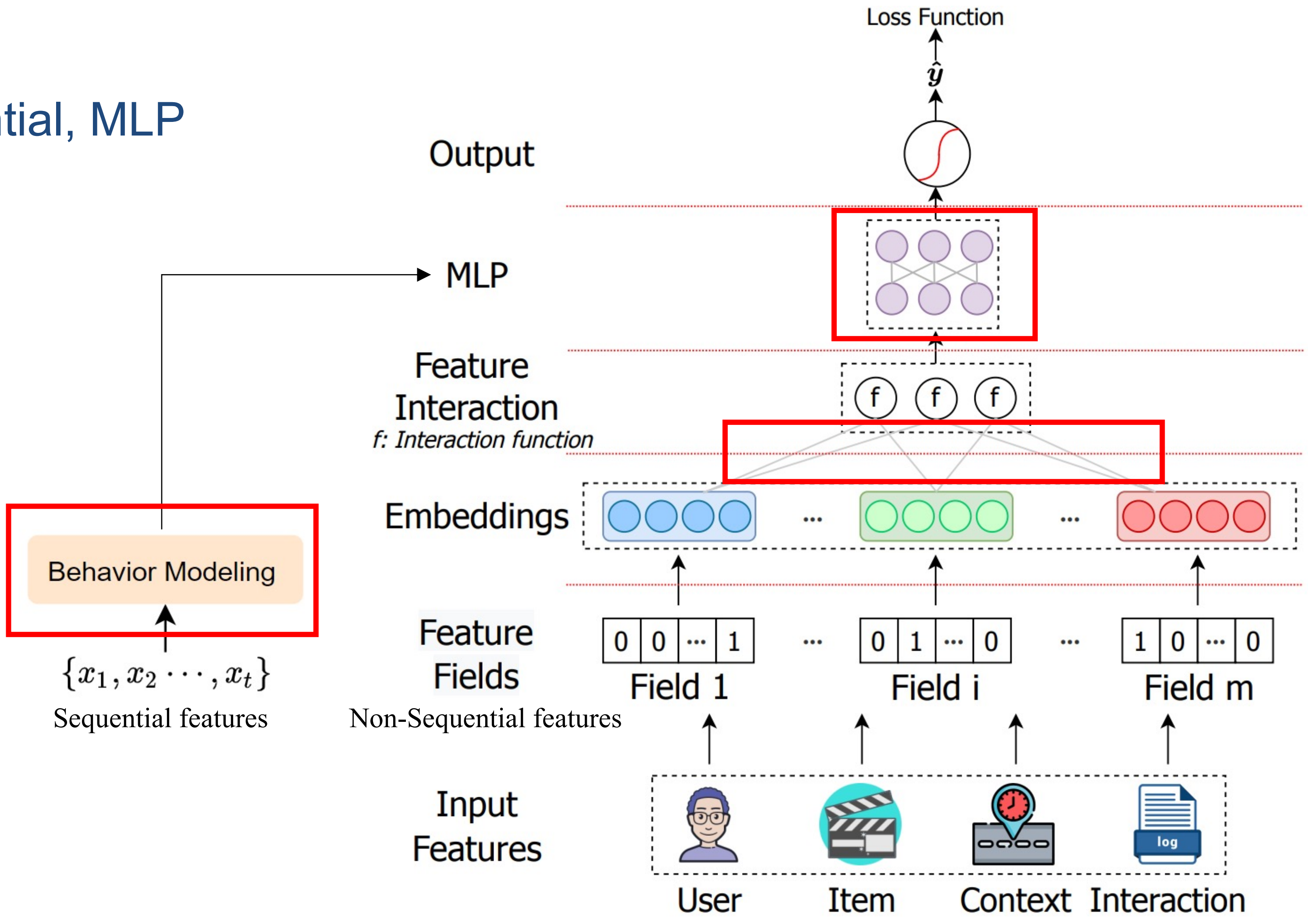
*More related work please refer to our survey: https://arxiv.org/pdf/2204.01390.pdf

Table of Contents

AMEIR

- Motivation:
  - 3 parts: sequential, non-sequential, MLP
  - Unified model for all scenarios
  - Restricted search space

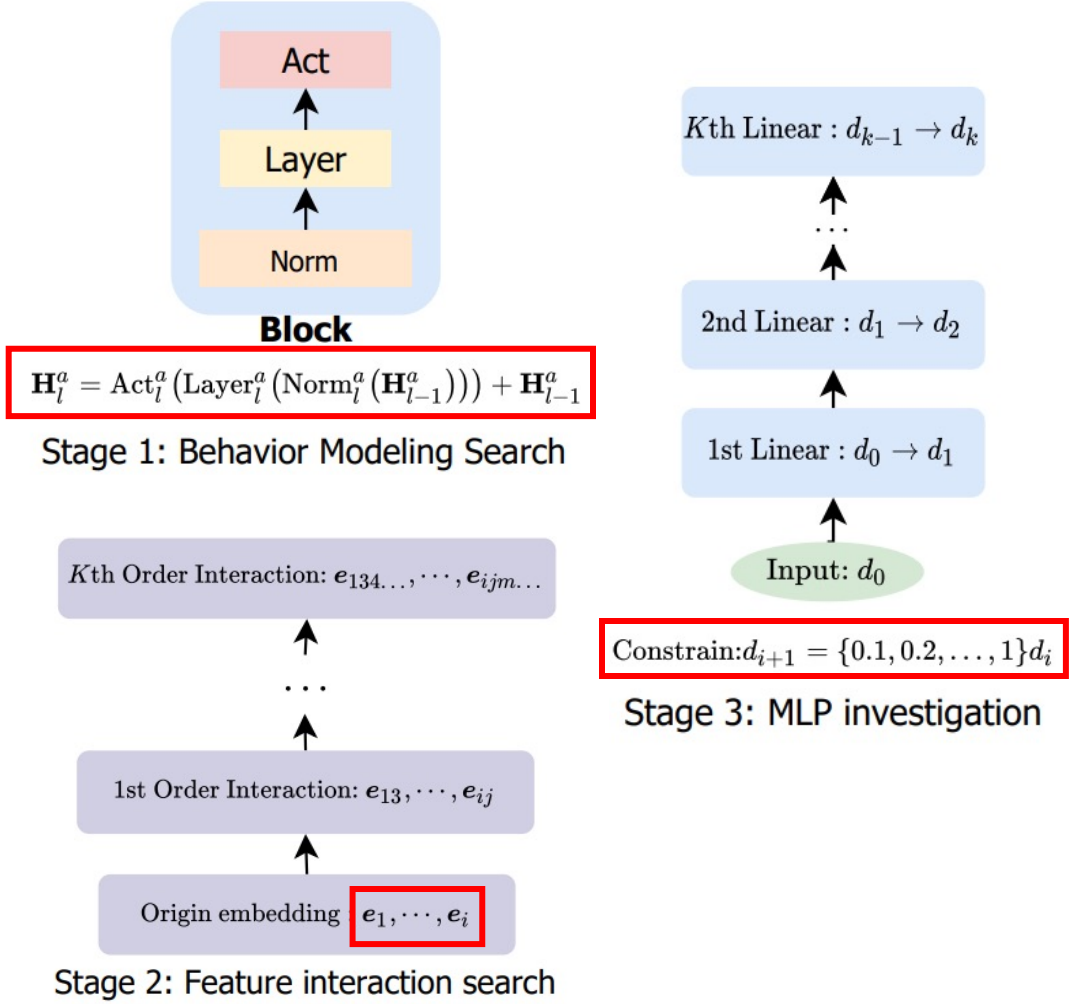
- Target:
  - Searching for 3 parts
  - Adaptive model



AMEIR: Automatic Behavior Modeling, Interaction Exploration and MLP Investigation in the Recommender System
IJCAI, 2021

AMEIR – Search Space

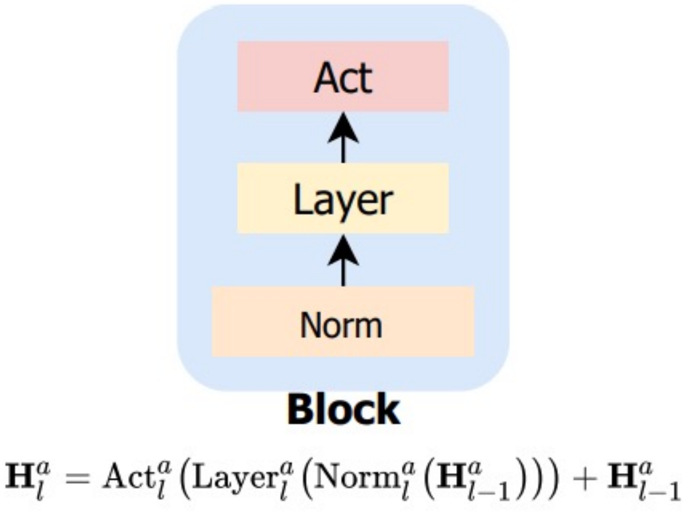
- Subspace 1 (Behavior modeling)
  - Searching for a fixed number of layers (L)
  - Normalization {Layer normalization, None}
  - Layer {Conv, Recur, Pooling, Attention}
  - Activation {ReLU, GeLU, Swish, Identity}
- Subspace 2 (Interaction exploration)
  - Interaction function: hadamard product (fixed)
  - Feature interaction candidates
- Subspace 3 (MLP investigation)
  - MLP dimension
  - Activation: {ReLU, Swish, Identity, Dice}



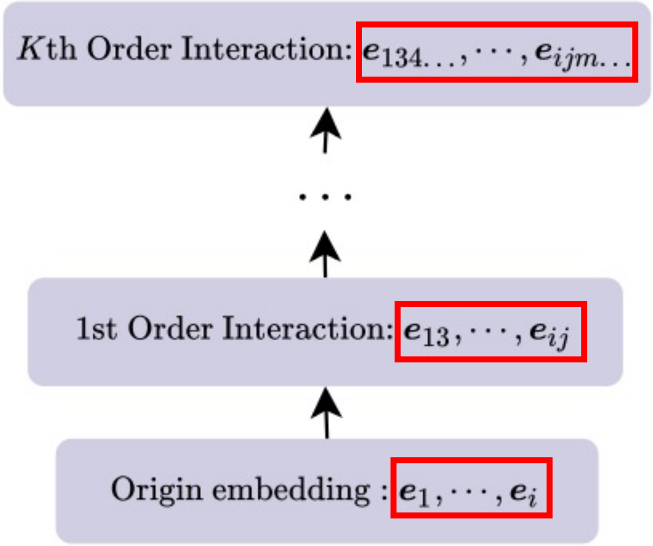
AMEIR: Automatic Behavior Modeling, Interaction Exploration and MLP Investigation in the Recommender System
IJCAI, 2021

AMEIR

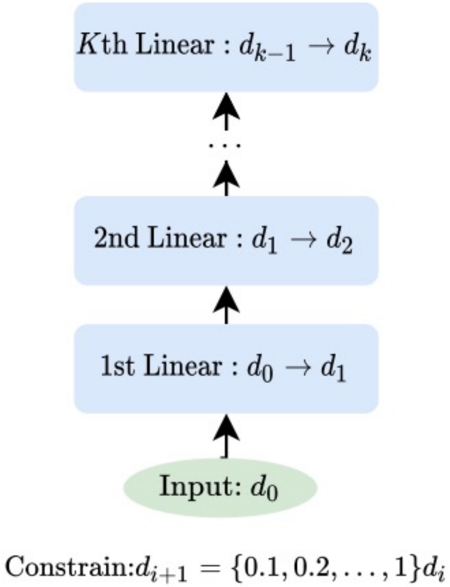
- Overall search strategy: One-shot random search
- Step 1: Using a predefined MLP, search for the optimal architecture.
- Step 2: Combined with SMBO, progressively expand the interaction sets, also use a predefined MLP.
- Step 3: Using a weight matrix of maximal dimension to realize one-shot search



Stage 1: Behavior Modeling Search



Stage 2: Feature interaction search

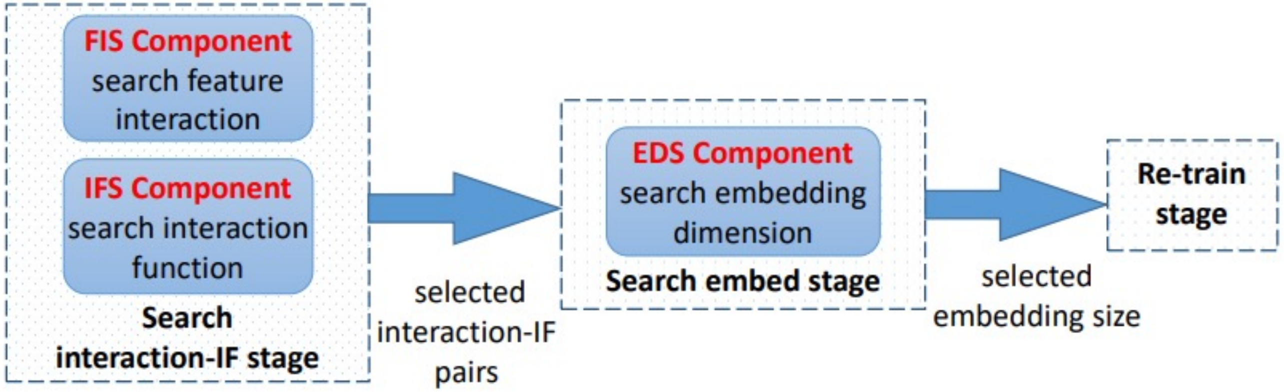


Stage 3: MLP investigation

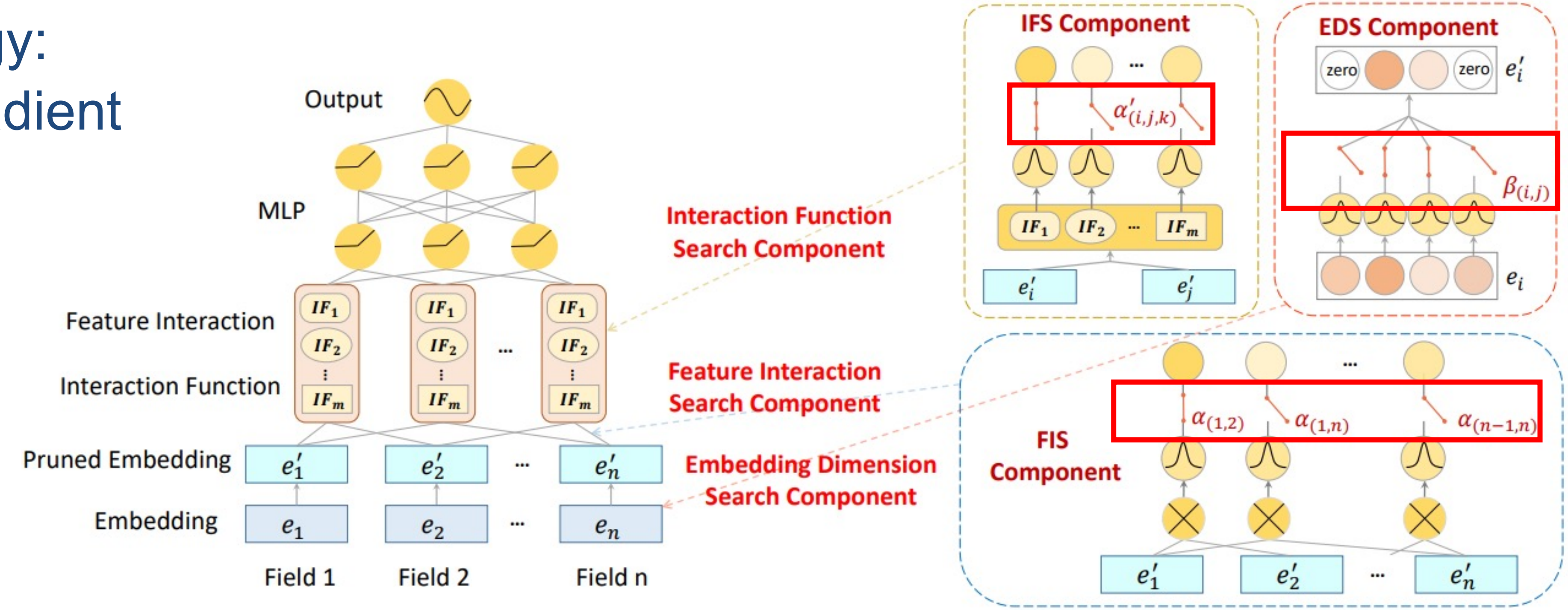
AMEIR: Automatic Behavior Modeling, Interaction Exploration and MLP Investigation in the Recommender System
IJCAI, 2021

AIM

- Search space:
  - Feature interaction
  - Interaction function
  - Embedding dimension

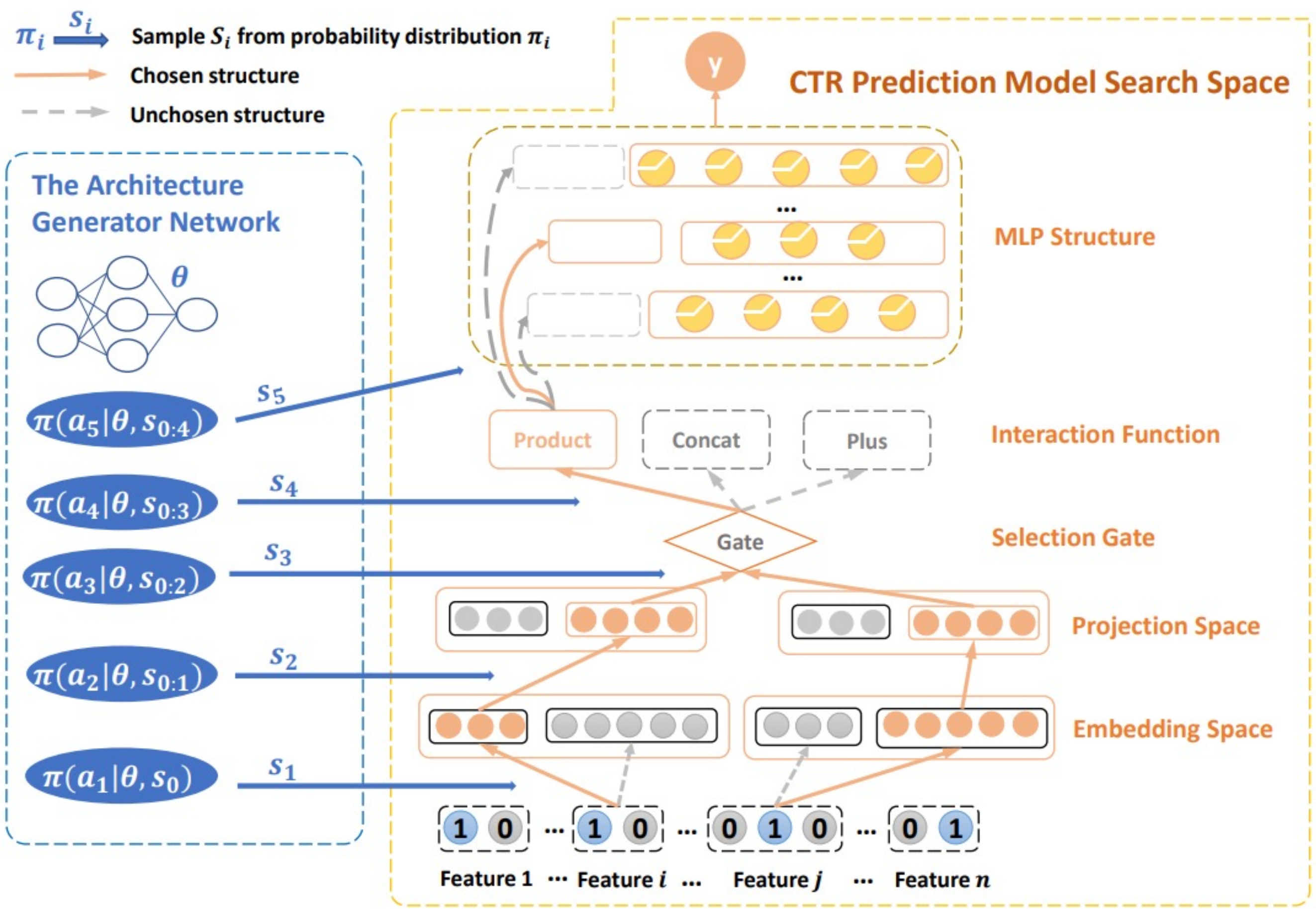


- Strategy:
  - Gradient



AIM: Automatic Interaction Machine for Click-Through Rate Prediction, TKDE

AutoIAS



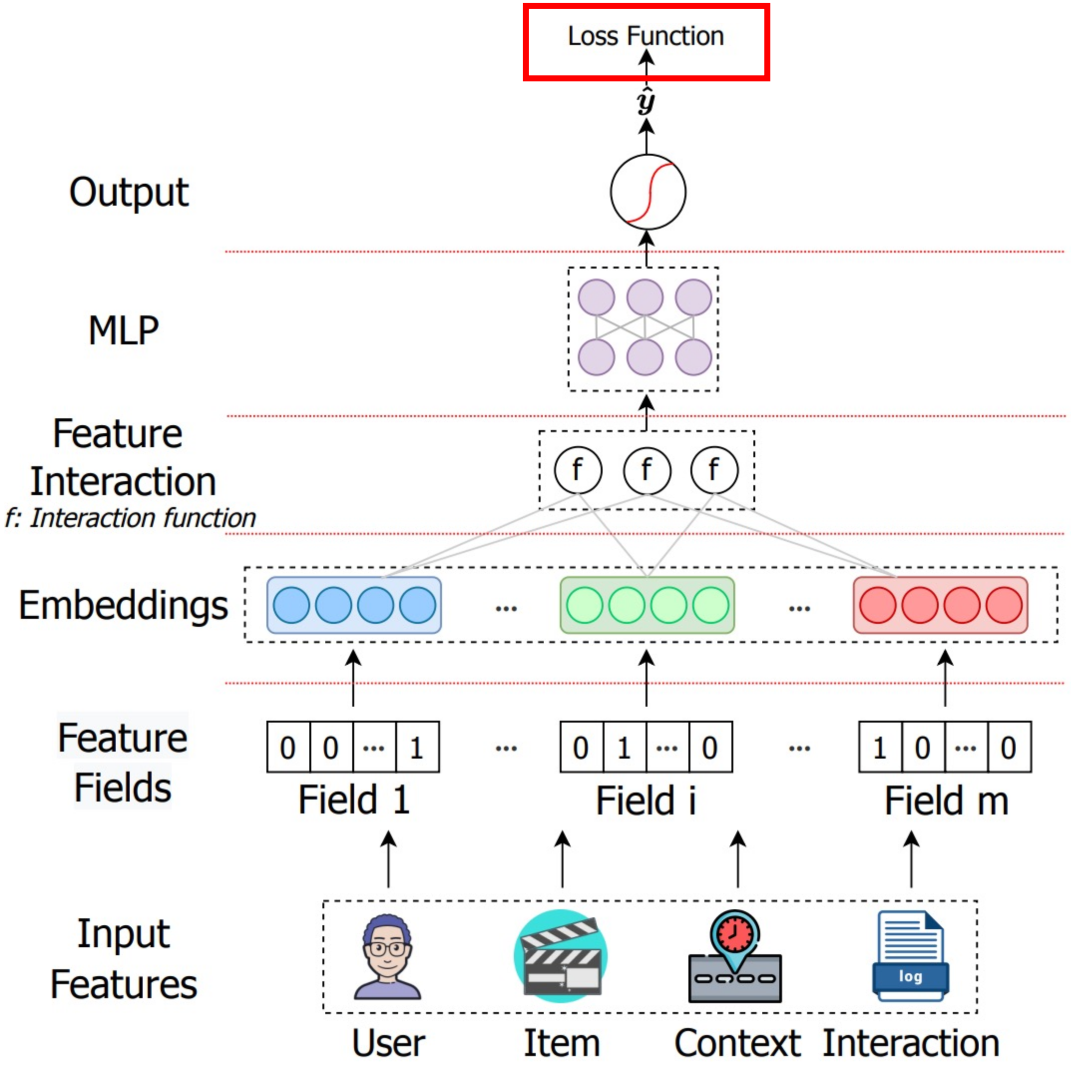
- Search space:
  - Embedding size
  - Projection size
  - Feature interaction candidates
  - Interaction function
  - MLP:
    - The number of layers
    - Layer dimensions
- Strategy:
  - Knowledge distillation
  - Reinforcement learning

AutoIAS: Automatic Integrated Architecture Searcher for Click-Trough Rate Prediction, CIKM, 2021

Table of Contents

AutoLoss

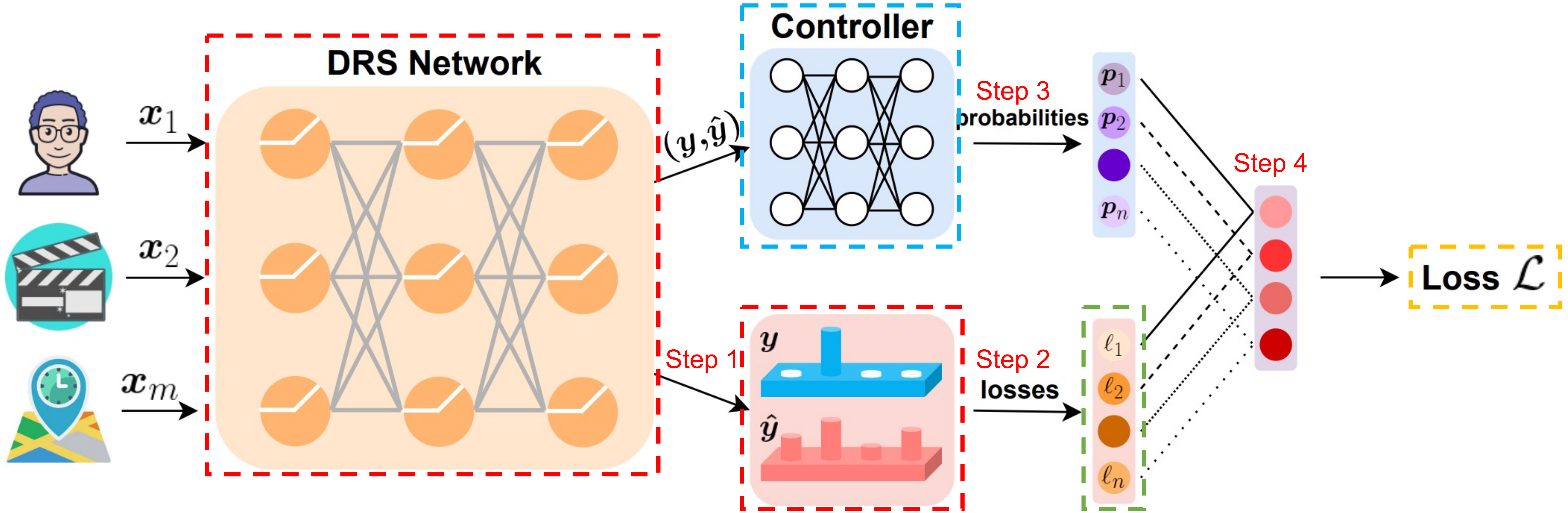
- Motivation:
  - Predefined and fixed loss function
  - Exhaustively or manually searched fused loss
- Target:
  - Searching for loss function
  - Considering convergence behavior



AutoLoss: Automated Loss Function Search in Recommendations, KDD, 2021
Author Slides Link: https://zhaoxyai.github.io/paper/kdd2021slides.pdf

AutoLoss – Forward-propagation

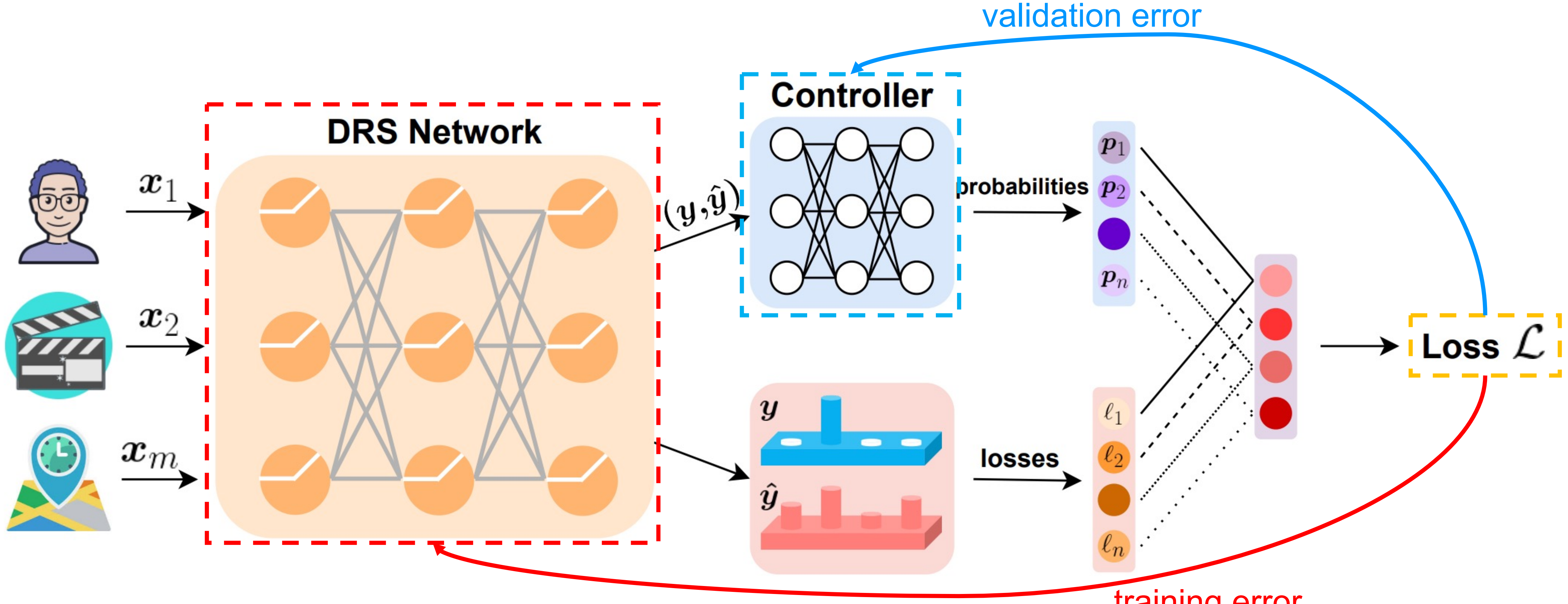
- Step 1: the DRS makes predictions
- Step 2: calculating candidate losses
- Step 3: the controller generates weights(probabilities) according to predictions
- Step 4: calculating the overall Loss (Weighted sum)



AutoLoss: Automated Loss Function Search in Recommendations, KDD, 2021

AutoLoss – Backward-propagation

- DRS network: updated based on training error
- Controller: updated based on validation error

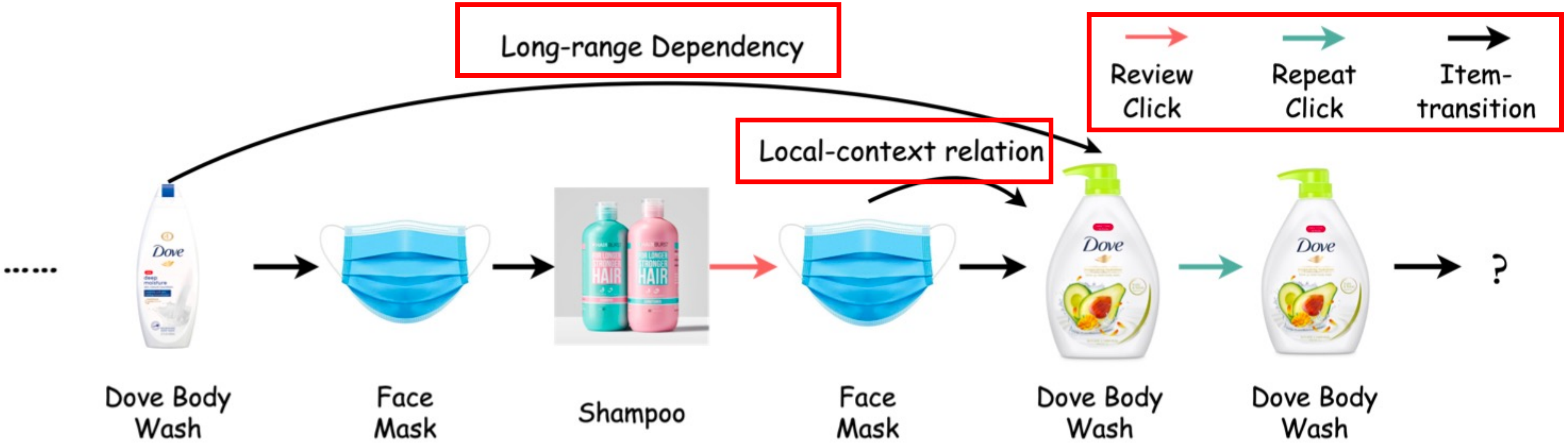


AutoLoss: Automated Loss Function Search in Recommendations, KDD, 2021

AutoGSR

- Target: searching for GNN architectures
- Motivation:
  - 3 kinds of information
  - 5 popular GNNs

Session graph	Aggregation function
EOP Multigraph	EOPA [3]
Shortcut Graph	SGAT [3]
Relational Graph	Relational GAT [30]
EOP Relational Graph	**Relational GGNN**
Mixup Graph	**Mixup**

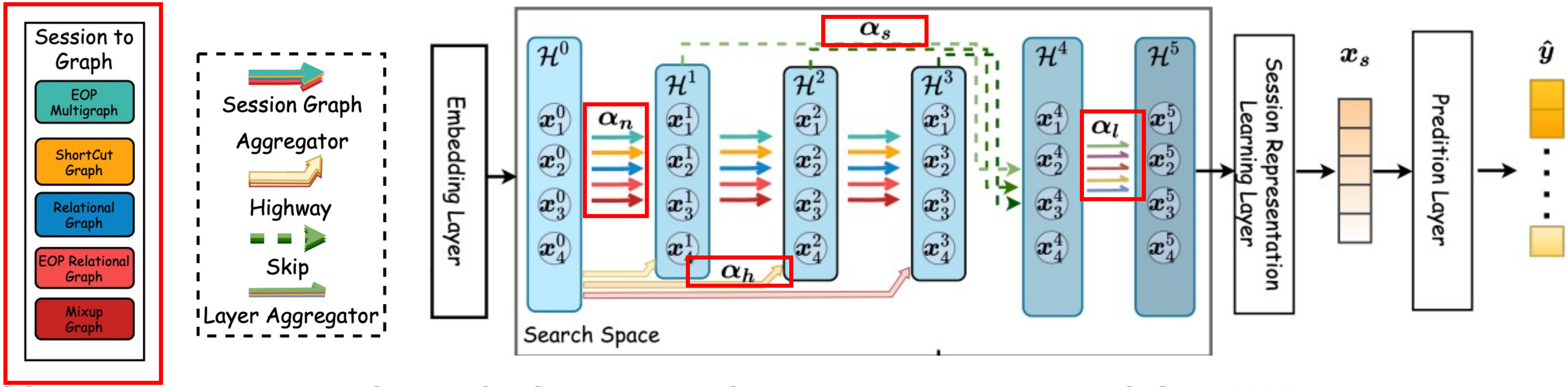


AutoGSR: Neural Architecture Search for Graph-based Session Recommendation, SIGIR, 2022

AutoGSR

- Search space:
  - Session aggregation: 5 popular graph types.
  - Layer aggregation: mean, max, concat, sum & highway & skip.

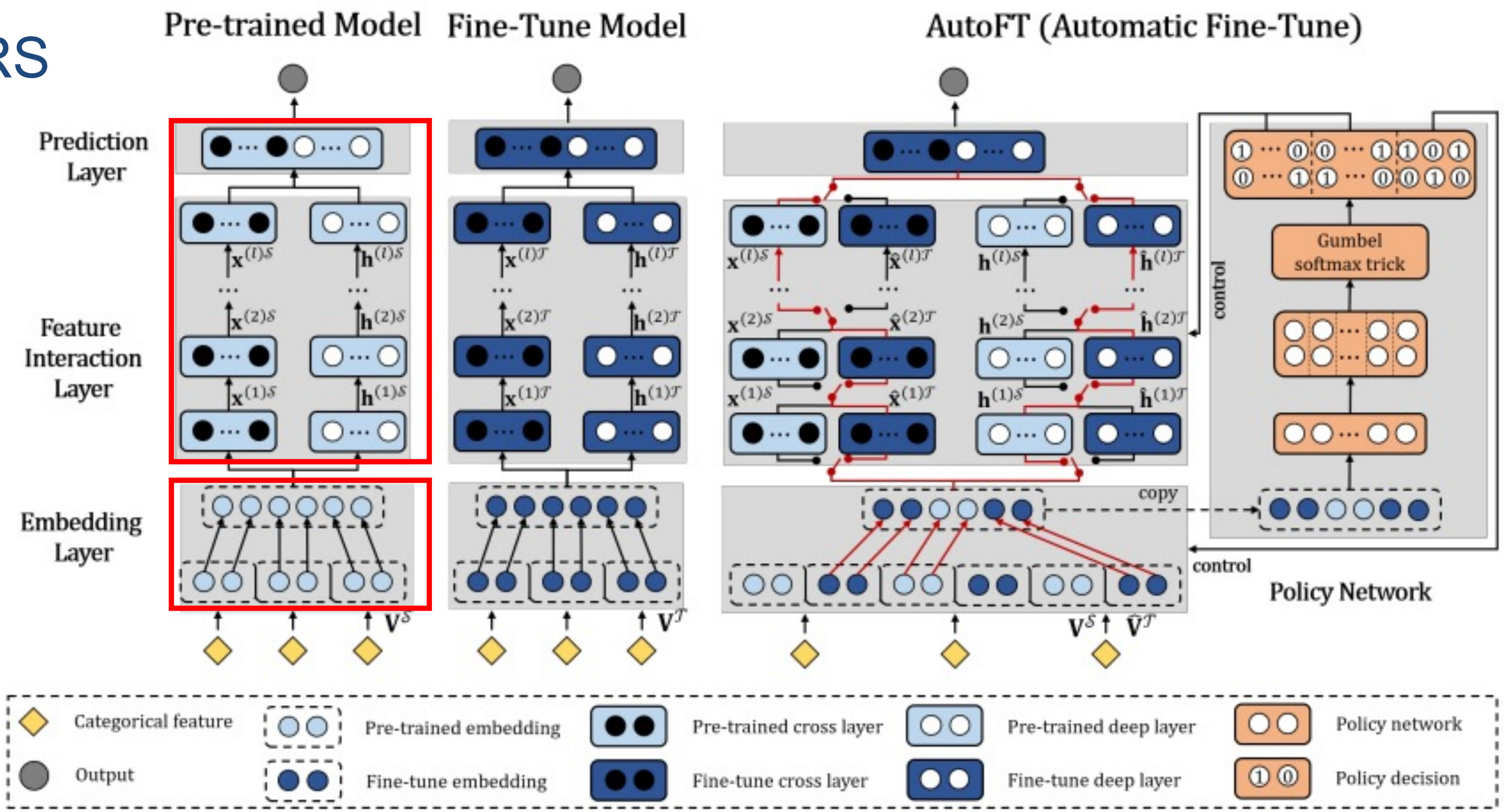
- Strategy: continuous relaxion & gradient



AutoGSR: Neural Architecture Search for Graph-based Session Recommendation, SIGIR, 2022

AutoFT

- Target: transfer learning DRS
- Search space:
  - Field-wise transfer
  - Layer-wise transfer
- Strategy: gradient



AutoFT: Automatic Fine-Tune for Parameters Transfer Learning in Click-Through Rate Prediction

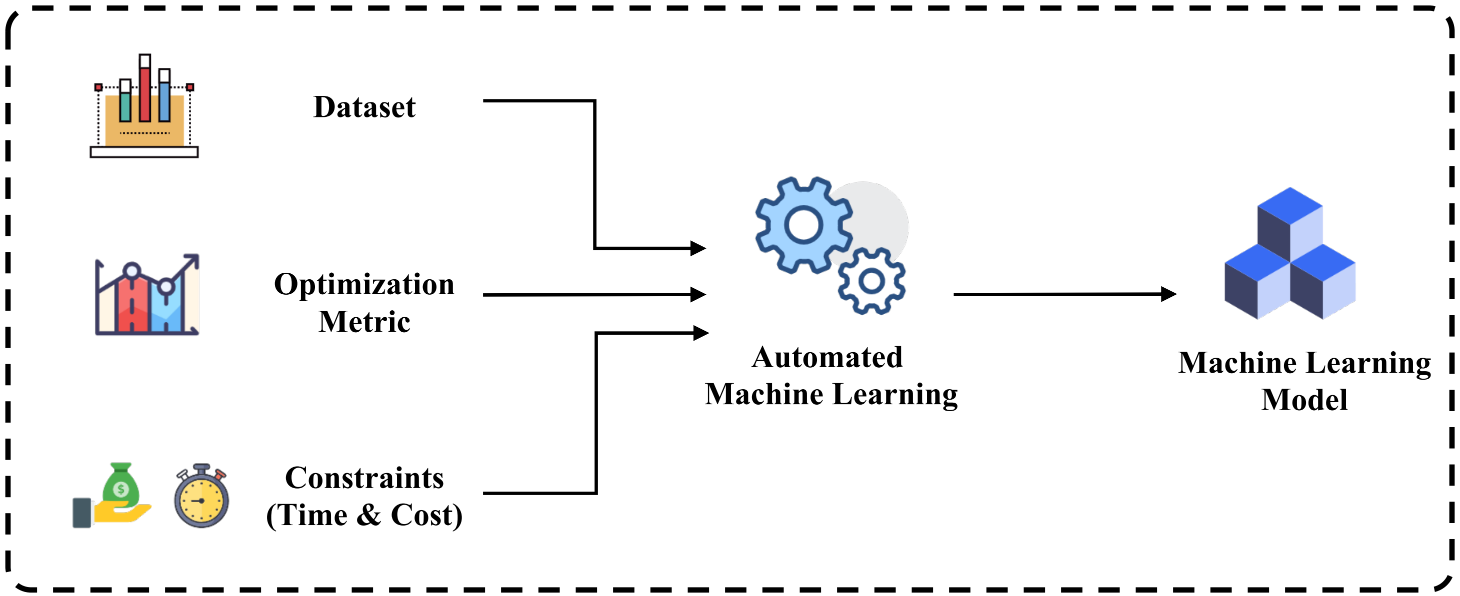
Table of Contents

Conclusion

Automated Machine Learning contribute to improving the performance of deep recommender systems in a data-driven manner.

- Search embedding dimensions to better model **feature representations**
- Design deep networks to better capture **feature interactions**
- Design **comprehensive system architectures** to better improve performance

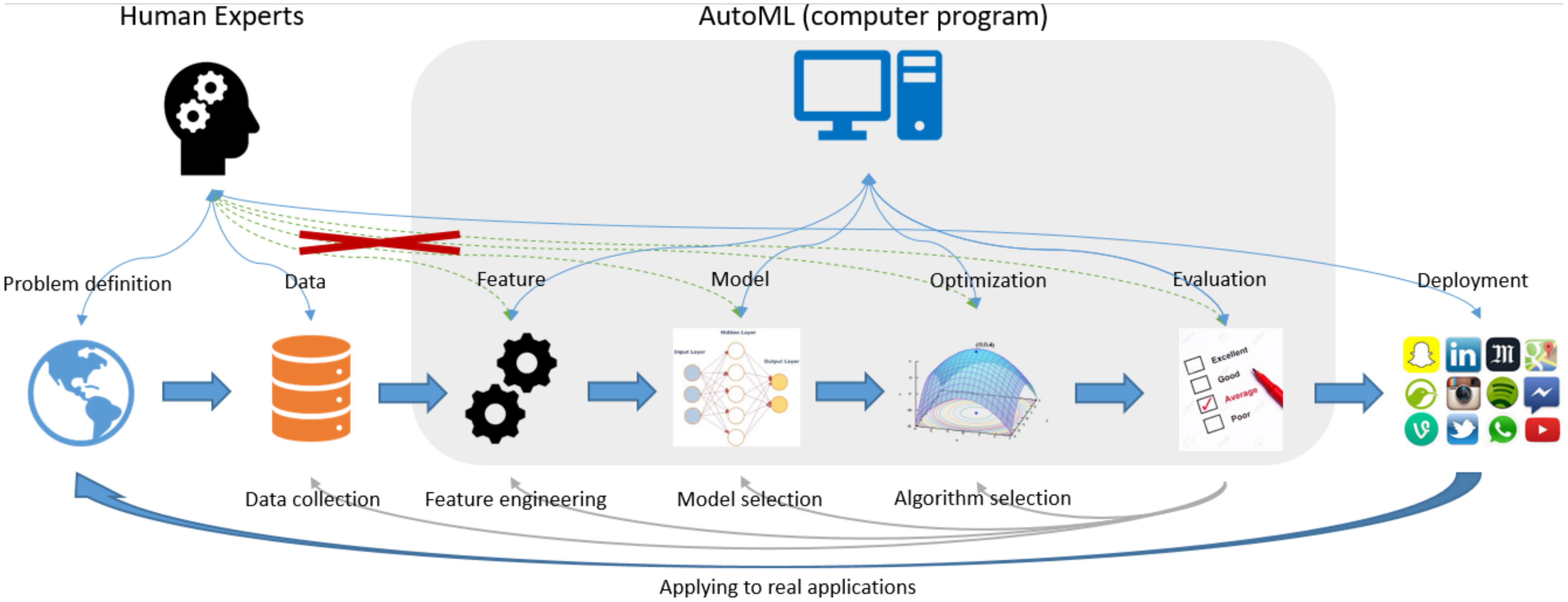
Automated Machine Learning



Conclusion

AutoML advantages:

- Different data → different architectures
- Less expert knowledge
- Saving time and efforts



Summarize DRS Embedding

	Model	Feature Field	Search Space	Search Strategy
Single Embedding Search	AMTL	Categorical	d^V	Gradient
	PEP	Categorical	2^{Vd}	Regularization
	AutoEmb	Categorical	a^V	Gradient
	ESPAN	Categorical	a^V	Reinforcement Learning
Group Embedding Search	AutoDim	Categorical	a^m	Gradient
	DNIS	Categorical	2^{bd}	Gradient
	NIS	Categorical	b^a	Reinforcement Learning
-	AutoDis	Numerical	2^{km}	Gradient

* d is the embedding size, V is the vocabulary size, m is the number of feature fields, a is the number of sub-dimensions, b is the number of groups, k is the number of meta-embeddings. (a < d, b<<V)

- The search space of Group Embedding Search is less than Single Embedding Search.
- Limited approaches for embedding learning of numerical features.
- Gradient-based is more popular as it has higher efficiency.

Summarize DRS Interaction

Type	Model	Search Space	Search Strategy
Feature Interaction Search	AutoFIS	$2^{C_m^p}$	Gradient
	AutoGroup	2^{gm}	Gradient
	FIVES	2^{m^2}	Gradient
Interaction Function Search	SIF	b^{a}	Gradient
	AutoFeature	$\mathrm{b}^{C_m^p}$	Evolutionary
Interaction Block Search	AutoCTR	b^{a}	Evolutionary
	AutoPI	b^{a}	Gradient

* m is the number of feature fields, p is the order, g is the number of pre-defined groups, a is the number of pre-defined bolcks, b is the number of candidate interaction functions.

- The search space of Feature Interaction Search and Interaction Function Search are larger than Interaction Block Search.
- Gradient-based is more popular as it has higher efficiency.

Summarize Comprehensive Search & System

Type	Model	Search Space	Search Strategy
Comprehensive Search	AMEIR	Sequential model, Feature interaction, MLP	One-shot Random Search
	AIM	Embedding Dimension, Intearaction Function, Feature Interaction	Gradient
	AutoIAS	Embedding Dimension, Projection Dimension, Interaction Function, Feature Interaction, MLP	Reinforcement Learning
System Design	AutoLoss	Optimization: Loss Function	Gradient
	AutoGSR	Structure Design: GNN Architecture	Gradient
	AutoFT	Parameter Tuning: Fine-Tune or Not (For pre-trained models)	Gradient

- Comprehensive search: separately search
- System design: AutoML is widely appied.
- Gradient-based is more popular as it has higher efficiency.

Future Directions

1) Feature Embedding

- Combine the feature representation learning with **model compression or quantization**
- **Multi-modality feature** representation learning, such as text, pictures, audio, and video

2) Feature Interaction

- **Personalized** feature interactions search for different users
- Introduce **complex interaction operators** for generating more diverse interaction functions

3) Comprehensive system architectures

- Searches for **multiple components** (embedding, interaction and MLP) **simultaneously**

Future Directions

4) AutoML Algorithm

- Design **AutoML algorithm (search and evaluation strategy)** that is more in line with the recommendation scenario

5) Model Selection

- Search different **sub-models/sub-architectures** according to different requests adaptively

6) Muti-task learning

- Multi-task learning is one of the most important techniques in industry recommendation for considering different revenue targets (e.g., ctr, cvr, vv). Designing an automatic algorithm for recommendation based on **muti-task learning**

7) User Behavior Modeling

- User history behaviors contain different dimensions of interests. Automatically retrieve **beneficial history behaviors** for modeling user preference

Q&A

Automated Machine Learning for Deep Recommender Systems: A Survey

https://arxiv.org/pdf/2204.01390