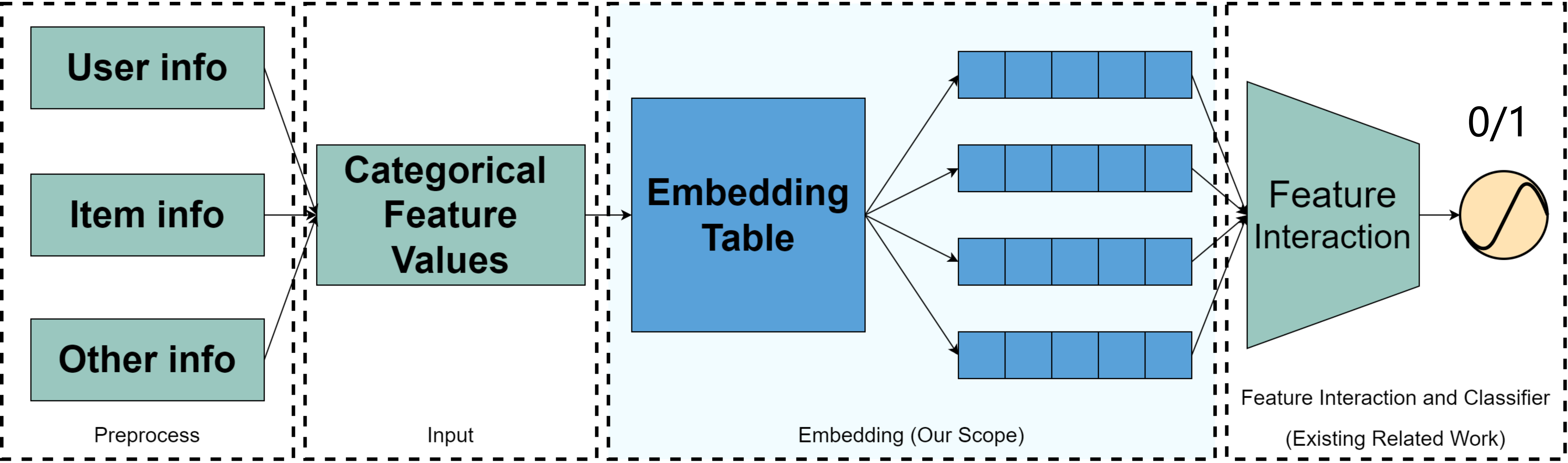
OptEmbed: Learning Optimal Embedding Table for Click-through Rate Prediction

Fuyuan Lyu[1,2*], Xing Tang[2*], Hong Zhu[2], Huifeng Guo[2], Yingxue Zhang[3], Ruiming Tang[2], Xue Liu[1]

[1]School of Computer Science, McGill University&MILA, Montreal, Canada
[2]Huawei Noah's Ark Lab, Shenzhen, China
[3]Huawei Noah's Ark Lab, Montreal, Canada

Presenter: Fuyuan Lyu

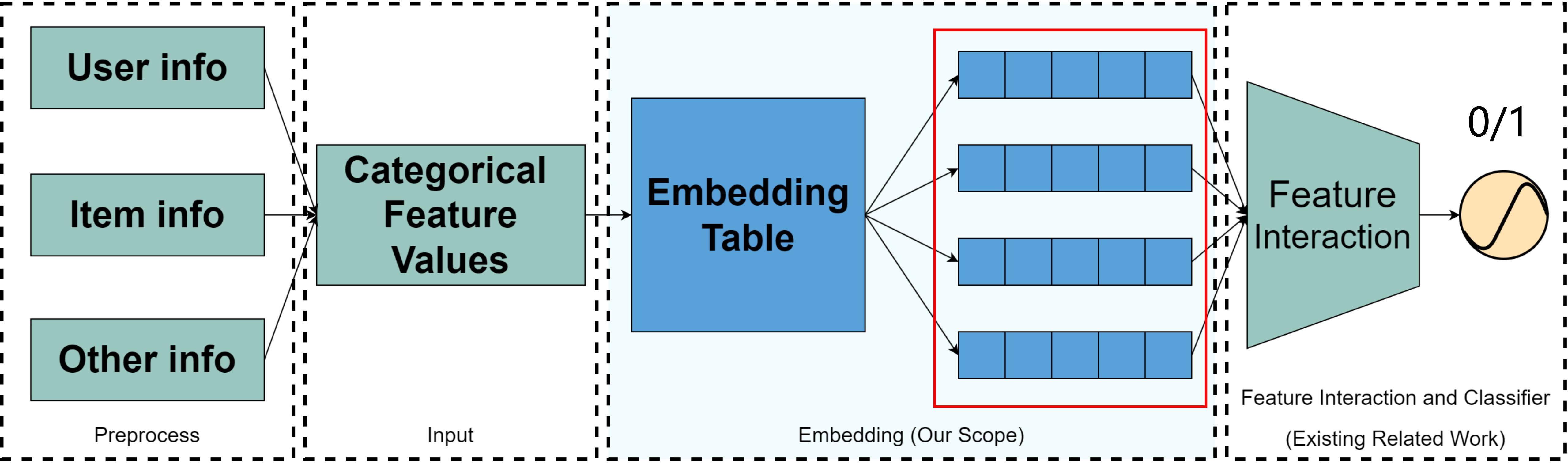
Background



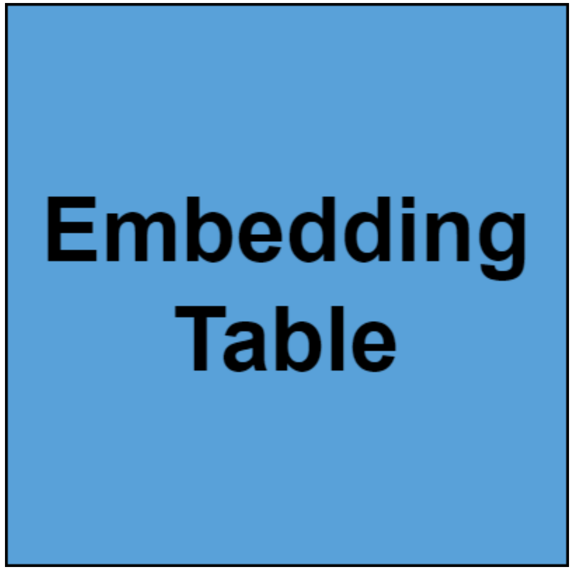
One-hot, sparse

Real-value, dense

Background

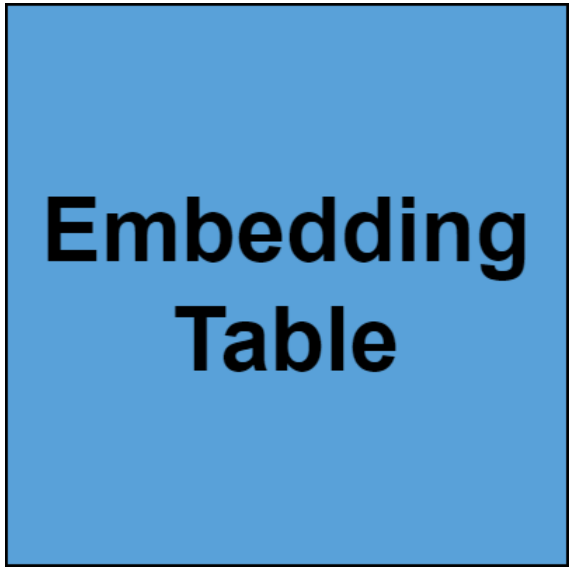


Is this optimal in terms of performance and efficiency?



A 2d tensor $E \in R^{|f| \times D}$
Feature size $|f| \approx 10^7$
Embed Dim $D \approx 64$

The Definition of “Optimal”



A 2d tensor $E \in R^{|f| \times D}$
Feature size $|f| \approx 10^7$
Embed Dim $D \approx 64$

1. **N**o **R**edundant **F**eatures in $|f|$.

   Redundant feature both consumes additional memory and is detrimental to model performance [1].

2. **E**mbedding **D**imension **F**lexible in D.

   Feature with small cardinality may induce overfitting, while feature with large cardinality may induce underfitting [2,3].

3. **H**ardware **F**riendly

   Sparse embedding table requires extra storage and decoding time, which is not suitable in practice [4,5].

[1] Wang, Yejing, et al. "Autofield: Automating feature selection in deep recommender systems." *Proceedings of the ACM Web Conference 2022*. 2022.
[2] Shi, Hao-Jun Michael, et al. "Compositional embeddings using complementary partitions for memory-efficient recommendation systems." *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 2020.
[3] Zhao, Xiangyu, et al. "Autodim: Field-aware embedding dimension searchin recommender systems." *Proceedings of the Web Conference 2021*. 2021.
[4] Deng, Wei, et al. "DeepLight: Deep lightweight feature interactions for accelerating CTR predictions in ad serving." *Proceedings of the 14th ACM international conference on Web search and data mining*. 2021.
[5] Liu, Siyi, et al. "Learnable Embedding sizes for Recommender Systems." *International Conference on Learning Representations*. 2020.

Comparison with Other Methods

Table 1: Comparison of embedding learning approaches.

Approach	R1: N.R.F.	R2: E.D.F.	R3: H.F.
MDE [9]	✗	✓	✓
DNIS [5]	✗	✓	✓
AutoDim [43]	✗	✓	✓
AutoField [35]	✓	✗	✓
QR [32]	✓	✗	✓
PEP [23]	✓	✓	✗
OptEmbed	✓	✓	✓

N.R.F, *E.D.F.* and *H.F.* are abbreviations for No Redundant Feature, Embedding Dimension Flexible and Hardware Friendly.

No previous work satisfies all three requirements!

CTR Prediction Formulation

For n field one-hot encoded raw input:

$$x = [x_{(1)}, x_{(2)}, \ldots, x_{(n)}]$$

Transform into dense vector:

$$e_{(i)} = E \times x_{(i)}$$

Concatenate all embeddings:

$$e = [e_{(1)}, e_{(2)}, \ldots, e_{(n)}] = E \times x$$

Fed into feature interaction and classification layer:

$$\hat{y} = \mathcal{F}(\mathrm{E} \times x | \mathrm{W})$$

Adopt the cross-entropy loss:

$$\mathrm{CE}(y, \hat{y}) = y \log(\hat{y}) + (1 - y) \log(1 - \hat{y}).$$

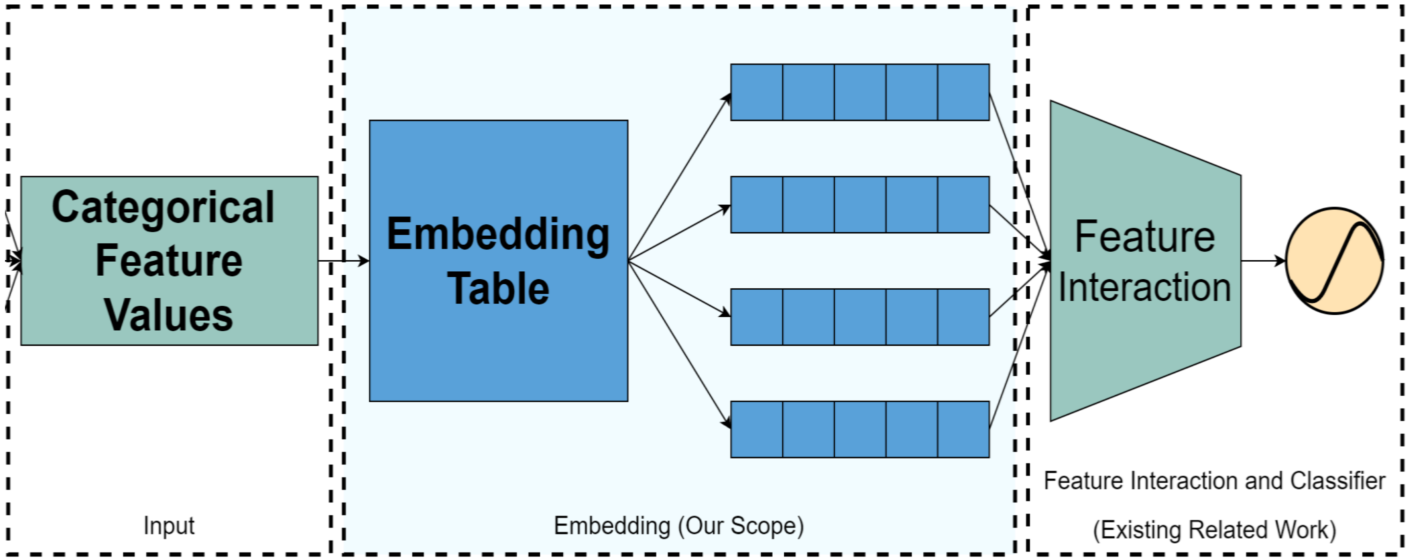


illustration figure for common CTR model

Formulate the CTR prediction problem: **Goal A**

$$\min_{\mathrm{E},\mathrm{W}} \mathcal{L}_{\mathrm{CE}}(\mathcal{D}|\{\mathrm{E}, \mathrm{W}\}) = -\frac{1}{|\mathcal{D}|} \sum_{(\mathrm{x}, y) \in \mathcal{D}} \mathrm{CE}(y, \mathcal{F}(\mathrm{E} \times \mathrm{x} | \mathrm{W}))$$

Optimal Embedding Table

Decompose the original single embedding table into a series of field-wise embedding table:

$$E = [E_{(1)}, E_{(2)}, \dots, E_{(n)}], E_{(i)} \in \mathrm{R}^{|f_{(i)}| \times D_{(i)}}$$

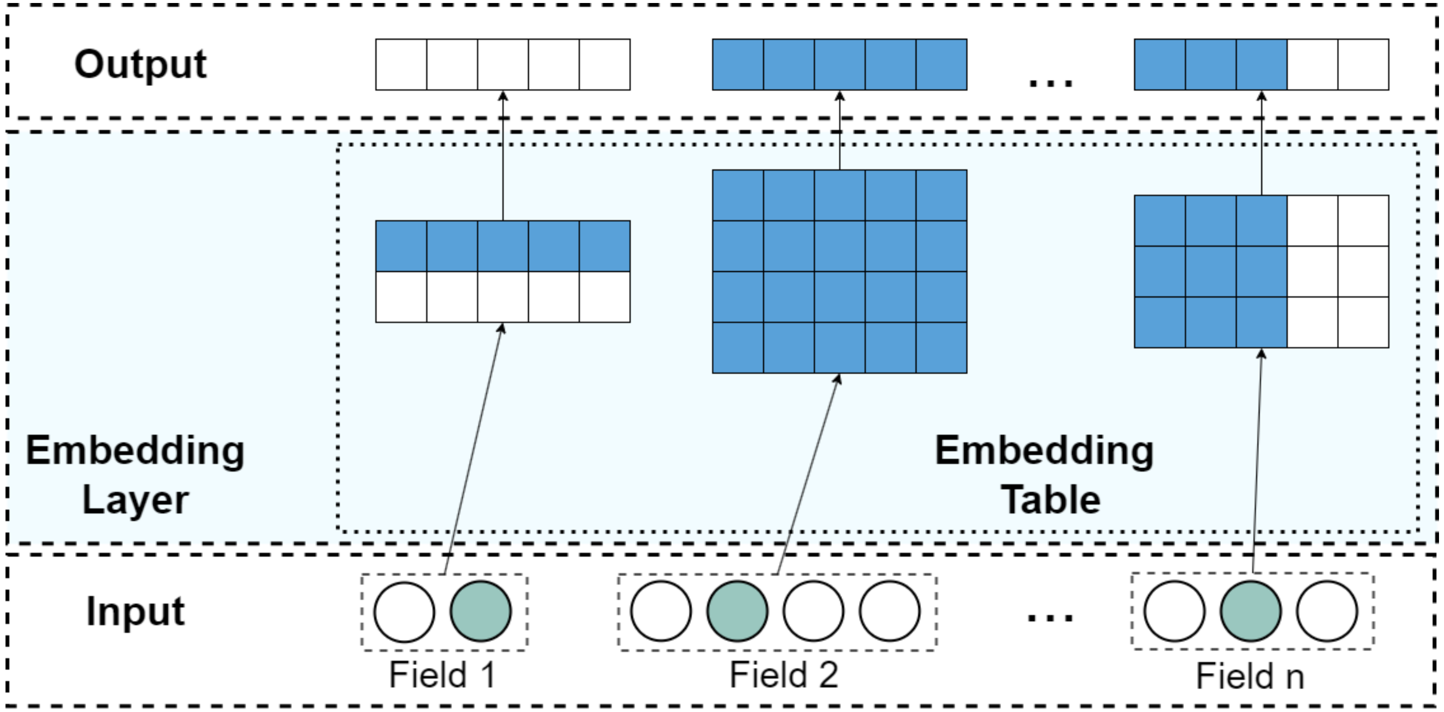
For **R1**: **N**o **R**edundant **F**eatures:

$$\sum_{i=1}^{n} |f_{(i)}| \leq |f|$$

R2: **E**mbedding **D**imension **F**lexible:

Choose different $D_{(i)}$

R3: **H**ardware **F**riendly is naturally satisfied.



Re-formulate the CTR prediction problem: **Goal B**

$$\min_{\mathrm{E}^*, \mathrm{W}} \mathcal{L}_{\mathrm{CE}}(\mathcal{D} | \{\mathrm{E}^*, \mathrm{W}\}), \ \mathrm{E}^* = [\mathrm{E}_{(1)}, \mathrm{E}_{(2)}, \cdots, \mathrm{E}_{(n)}],$$

$$s.t. \ \mathrm{E}_{(i)} \in \mathbb{R}^{|f_{(i)}| \times D_{(i)}}, \ \sum_{i=1}^{n} |f_{(i)}| \leq |f|, \ D_{(i)} \leq D, \ \forall i \leq n.$$

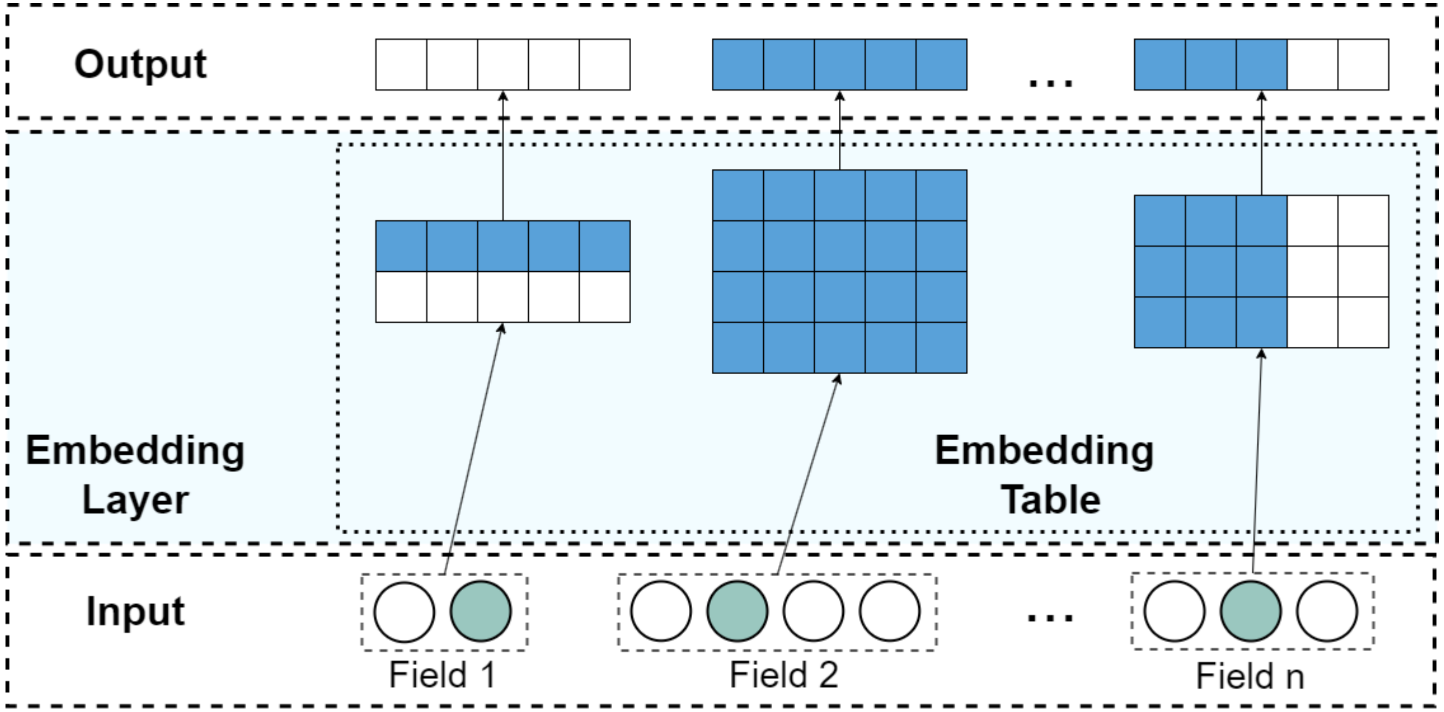
Optimal Embedding Table

Decompose the original single embedding table into a series of field-wise embedding table:

$$E = \left[E_{(1)}, E_{(2)}, \dots, E_{(n)}\right], E_{(i)} \in \mathrm{R}^{|\mathrm{f}_{(i)}| \times D_{(i)}}$$

For **R1**: **N**o **R**edundant **F**eatures:

$$\sum_{i=1}^{n} |f_{(i)}| \leq |f|$$



R2: **E**mbedding **D**imension **F**lexible:

Choose different $D_{(i)}$

R3: **H**ardware **F**riendly are naturally satisfied.

Re-formulate the CTR prediction problem: **Goal B**

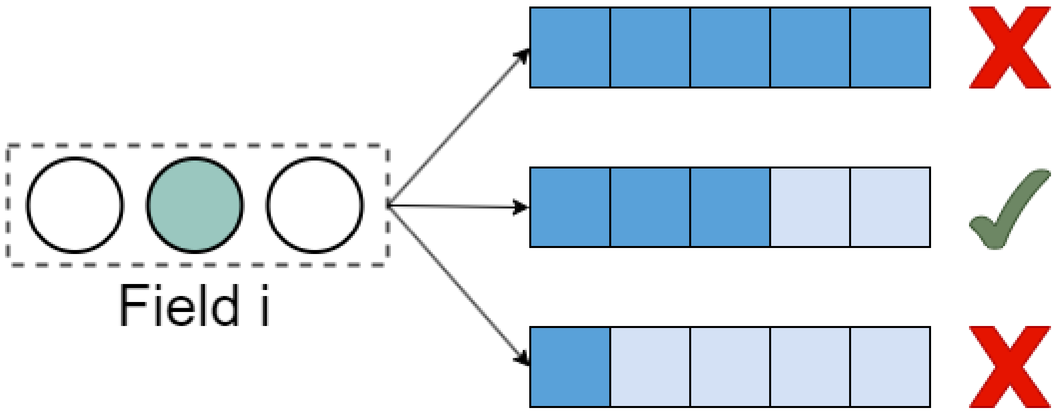
$$\min_{\mathrm{E}^*, \mathrm{W}} \mathcal{L}_{\mathrm{CE}}(\mathcal{D}|\{\mathrm{E}^*, \mathrm{W}\}), \ \mathrm{E}^* = [\mathrm{E}_{(1)}, \mathrm{E}_{(2)}, \cdots, \mathrm{E}_{(n)}],$$

$$\text{s.t. } \mathrm{E}_{(i)} \in \mathbb{R}^{|f_{(i)}| \times D_{(i)}}, \ \sum_{i=1}^{n} |f_{(i)}| \leq |f|, \ D_{(i)} \leq D, \ \forall i \leq n.$$

Hard to directly optimize!

Optimal Embedding Table

If optimize $|f_{(i)}|$ & $D_{(i)}$ alternately



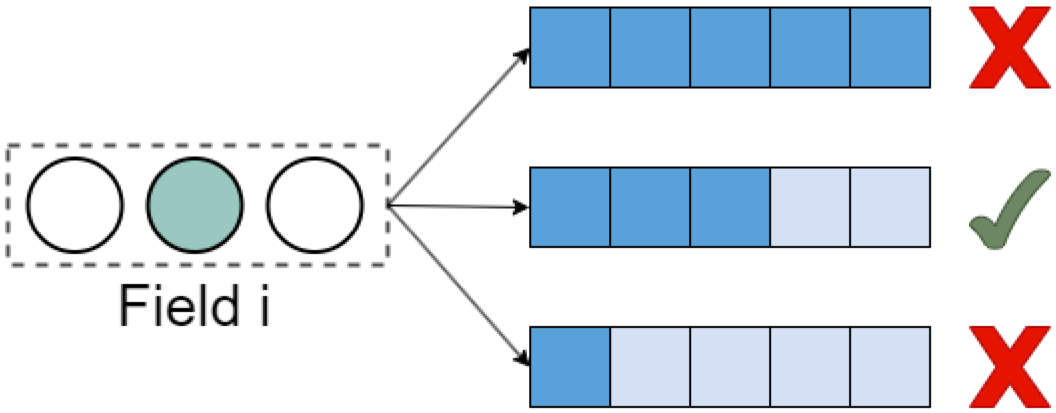
$|f_{(i)}|$ is influenced by $D_{(i)}$

Sub-optimal result

Hard to directly optimize!

Optimal Embedding Table

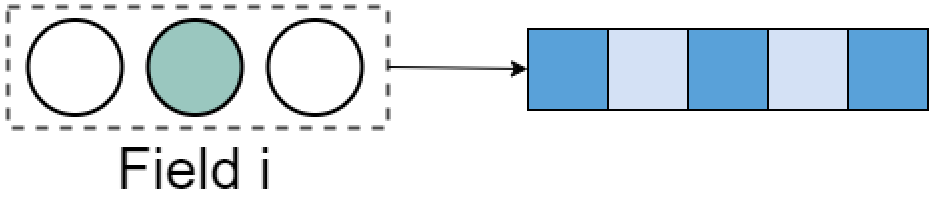
If optimize $|f_{(i)}|$ & $D_{(i)}$ alternately



$|f_{(i)}|$ is influenced by $D_{(i)}$

Sub-optimal result

If optimize $|f_{(i)}|$ & $D_{(i)}$ uniformly

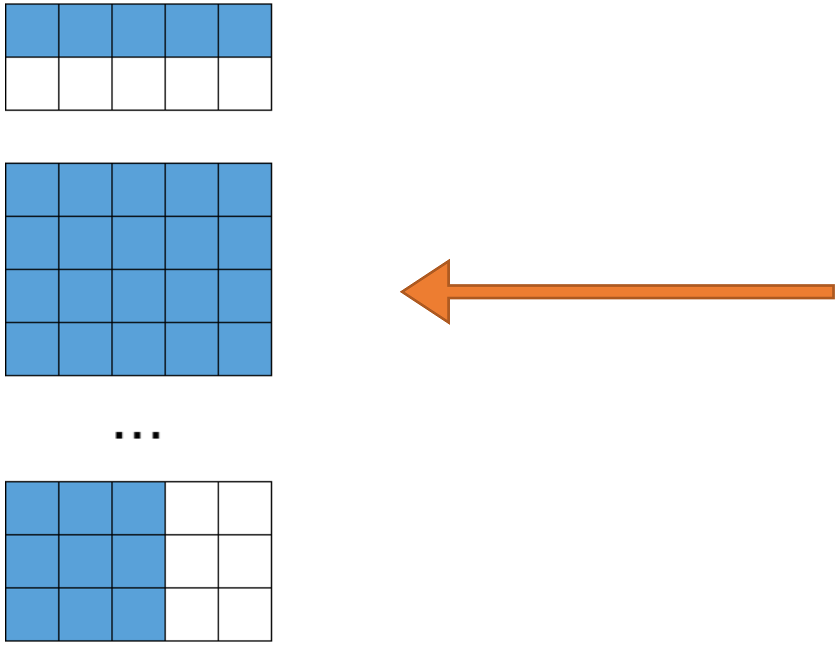


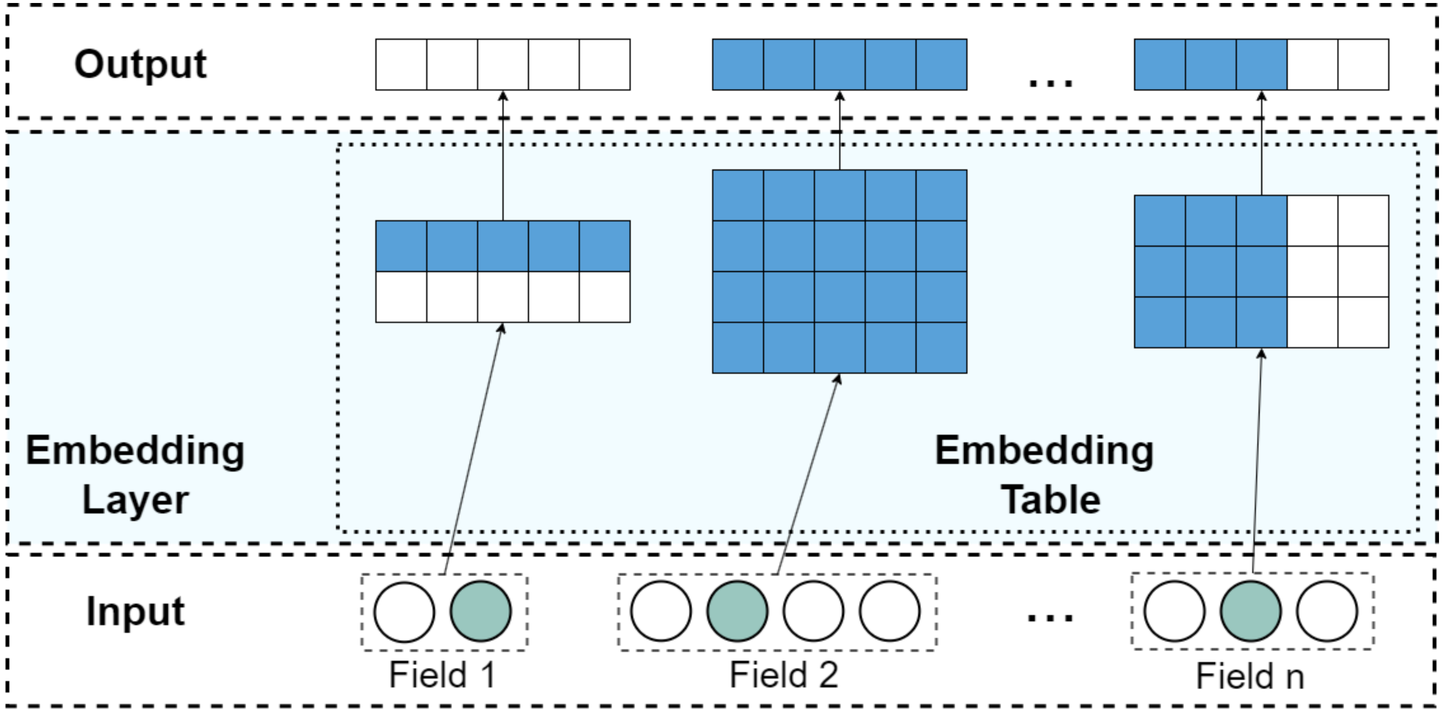
Sparse embedding

Hardware Unfriendly

Hard to directly optimize!

Optimal Embedding Table



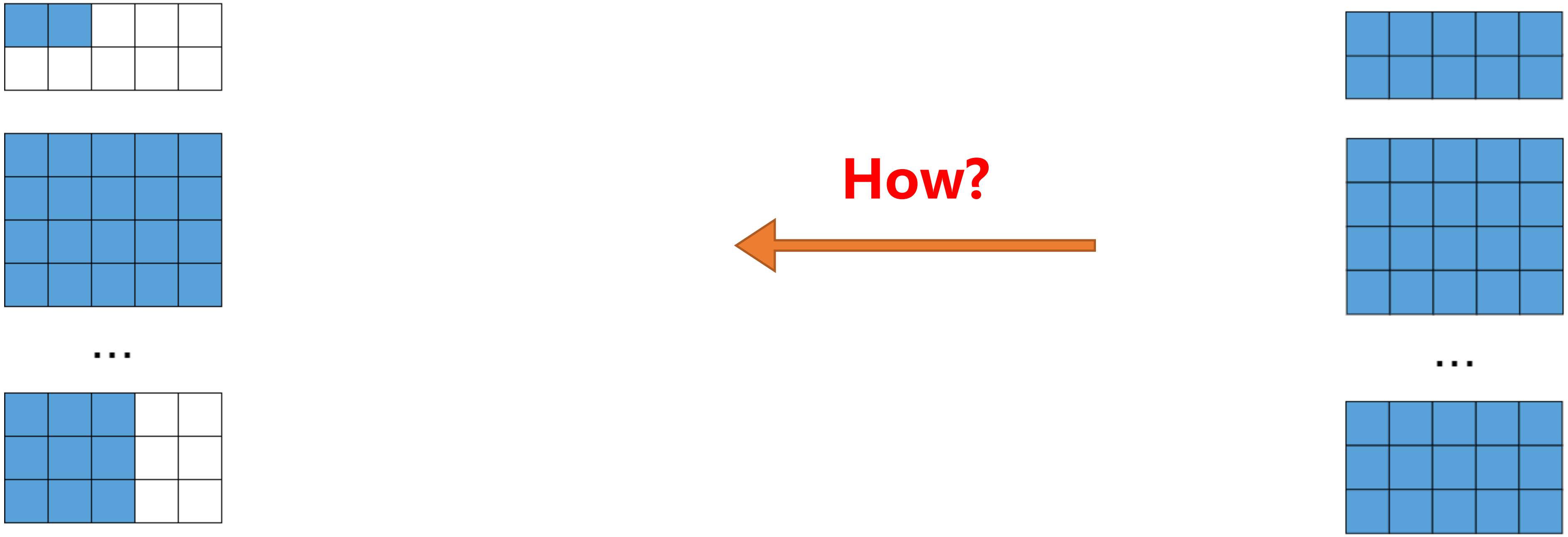


$$\min_{\mathrm{E}^*,\mathrm{W}} \mathcal{L}_{\mathrm{CE}}(\mathcal{D}|\{\mathrm{E}^*,\mathrm{W}\}),\ \mathrm{E}^* = [\mathrm{E}_{(1)},\mathrm{E}_{(2)},\cdots,\mathrm{E}_{(n)}],$$

$$s.t.\ \mathrm{E}_{(i)} \in \mathbb{R}^{|f_{(i)}|\times D_{(i)}},\ \sum_{i=1}^{n} |f_{(i)}| \leq |f|,\ D_{(i)} \leq D,\ \forall i \leq n.$$

Goal B

Optimal Embedding Table



$$\min_{\mathrm{E}^*,\mathrm{W}} \mathcal{L}_{\mathrm{CE}}(\mathcal{D}|\{\mathrm{E}^*,\mathrm{W}\}),\ \mathrm{E}^* = [\mathrm{E}_{(1)},\mathrm{E}_{(2)},\cdots,\mathrm{E}_{(n)}],$$

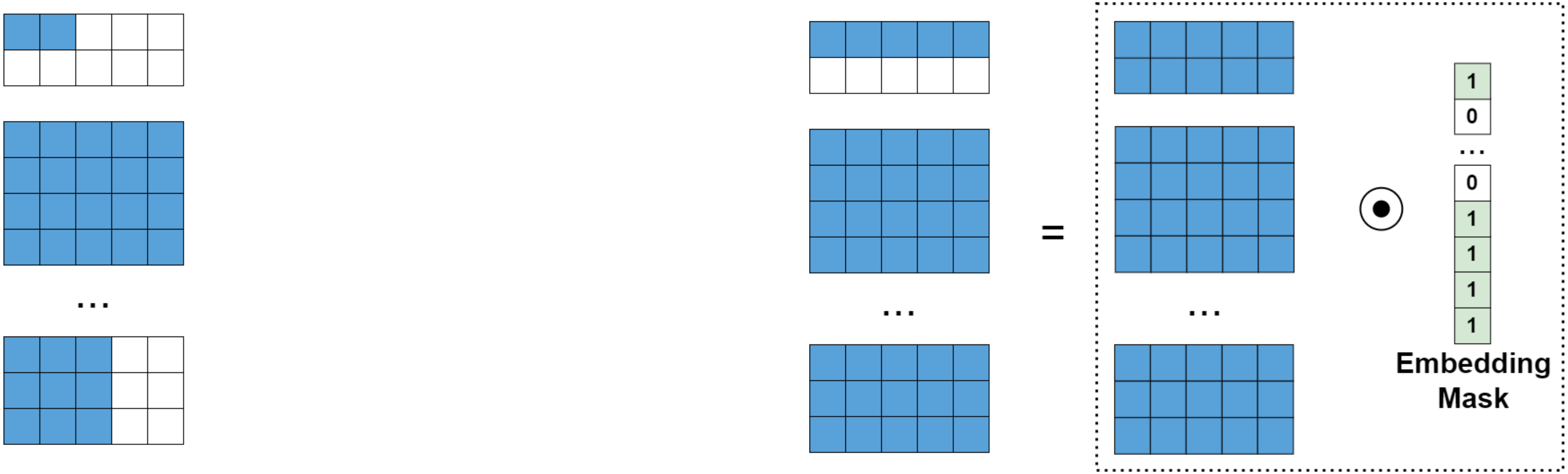
$$s.t.\ \mathrm{E}_{(i)} \in \mathbb{R}^{|f_{(i)}|\times D_{(i)}},\ \sum_{i=1}^{n} |f_{(i)}| \leq |f|,\ D_{(i)} \leq D,\ \forall i \leq n.$$

Goal B

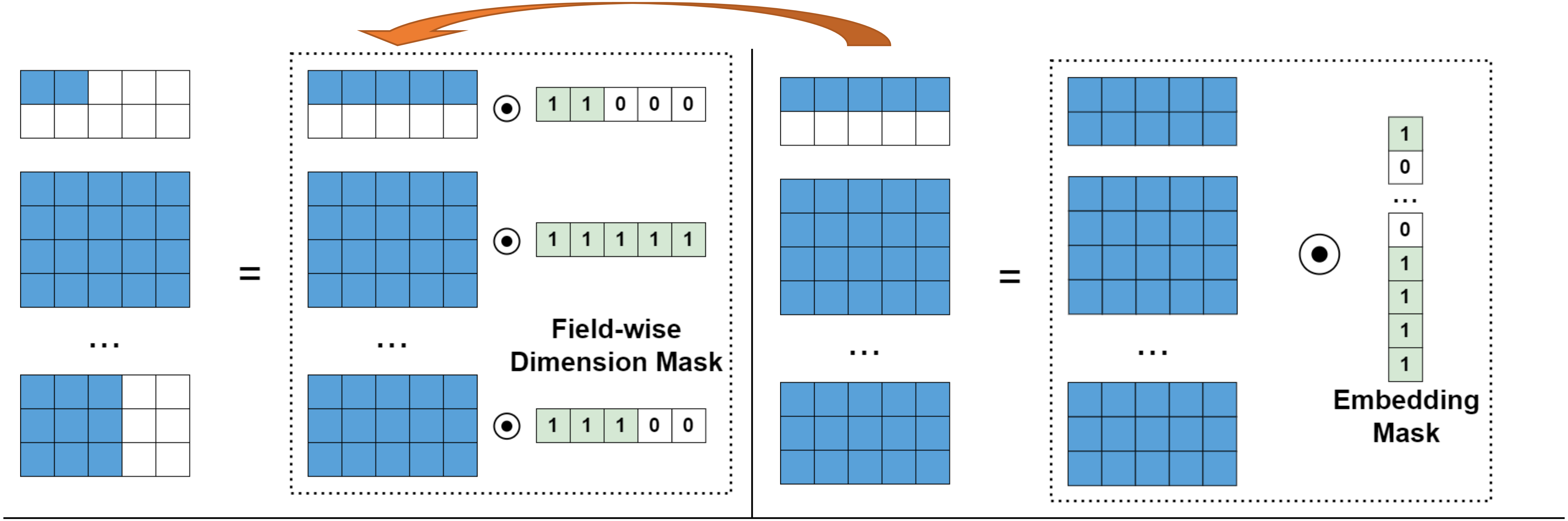
$$\min_{\mathrm{E},\mathrm{W}} \mathcal{L}_{\mathrm{CE}}(\mathcal{D}|\{\mathrm{E},\mathrm{W}\}) = -\frac{1}{|\mathcal{D}|}\sum_{(\mathrm{x},y)\in\mathcal{D}} \mathrm{CE}(y,\mathcal{F}(\mathrm{E}\times\mathrm{x}|\mathrm{W}))$$

Goal A

OptEmbed



OptEmbed



$$\min_{\mathrm{E}^*,\mathrm{W}} \mathcal{L}_{\mathrm{CE}}(\mathcal{D}|\{\mathrm{E}^*,\mathrm{W}\}),\ \mathrm{E}^* = [\mathrm{E}_{(1)},\mathrm{E}_{(2)},\cdots,\mathrm{E}_{(n)}],$$

$$s.t.\ \mathrm{E}_{(i)} \in \mathbb{R}^{|f_{(i)}|\times D_{(i)}},\ \sum_{i=1}^{n} |f_{(i)}| \leq |f|,\ D_{(i)} \leq D,\ \forall i \leq n.$$

$$\min_{\mathbf{m}_e,\mathbf{m}_d,\mathrm{E},\mathrm{W}} \mathcal{L}_{\mathrm{CE}}(\mathcal{D}|\{\mathrm{E}^*,\mathrm{W}\}),\ \mathrm{E}^* = \mathrm{E}\odot\mathbf{m}_e\odot\mathbf{m}_d.$$

Goal B **Goal C**

OptEmbed

In practice, we introduce two masks:

$$\min_{\mathbf{m}_e,\mathbf{m}_d,\mathbf{E},\mathbf{W}} \mathcal{L}_{\mathrm{CE}}(\mathcal{D}|\{\mathbf{E}^*,\mathbf{W}\}),\ \mathbf{E}^* = \mathbf{E} \odot \mathbf{m}_e \odot \mathbf{m}_d.$$

Goal C

Field-wise dimension mask:

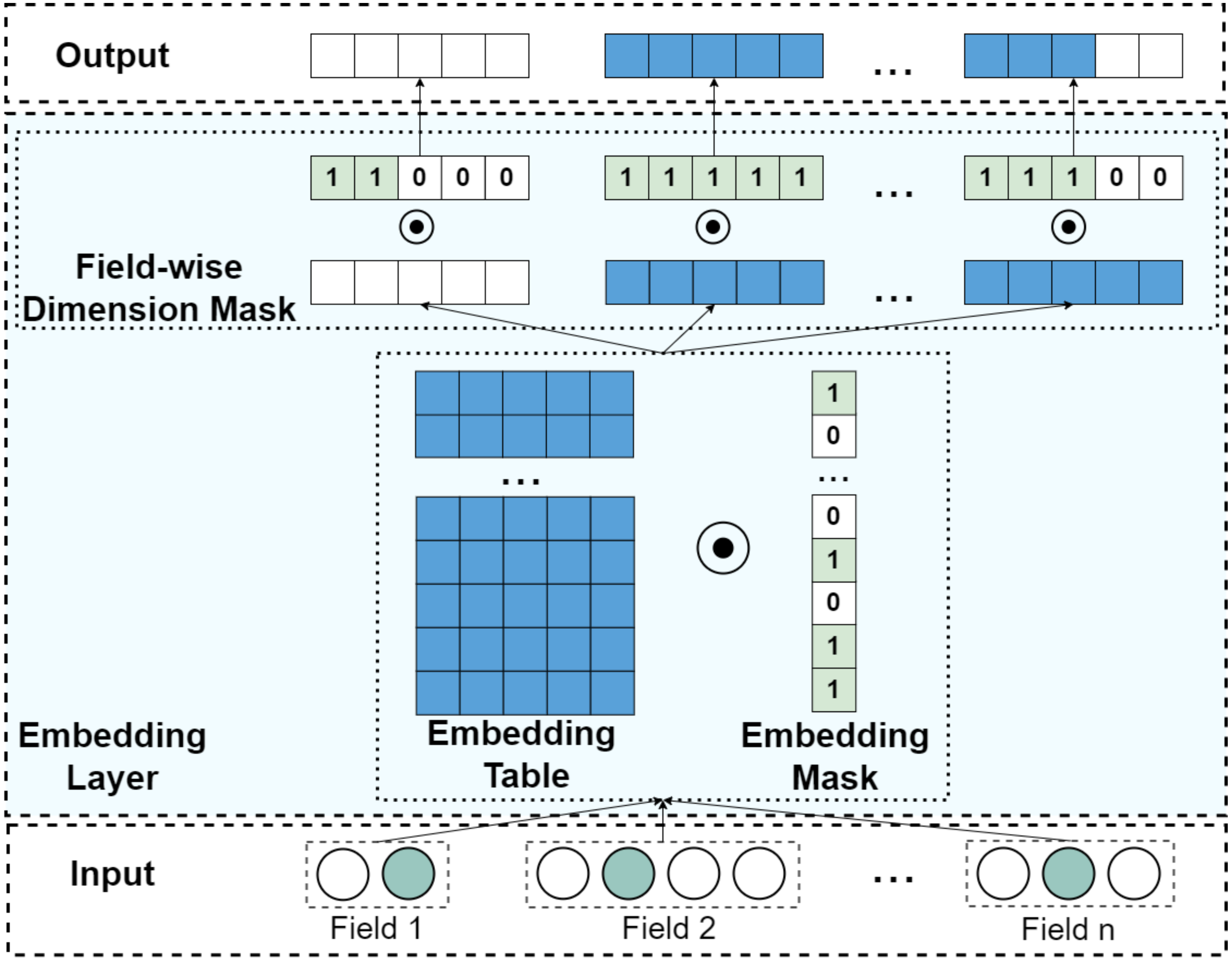
$$m_d \in \{0,1\}^{D\times n}$$

Embedding mask:

$$m_e \in \{0,1\}^{|f|}$$

Three phases:

1. Redundant Embedding Pruning
2. Embedding Dimension Search
3. Parameter Re-training



Experiment

Backbone Models:
DeepFM[1], DCN[2], FNN[3], IPNN[4]

Dataset	#samples	#field	#values	pos ratio
Criteo	4.6×10^7	39	6.8×10^6	0.23
Avazu	4.0×10^7	24	4.4×10^6	0.17
KDD12	1.5×10^8	11	6.0×10^6	0.06

Baseline Methods:
AutoDim[5], AutoField[6], QR[7], PEP[8]

Evaluation Metrics:
AUC, Logloss and Sparsity

$$\text{Sparsity} = 1 - \frac{\#\text{Remaining Params}}{|f| \times D}$$

[1] Guo, Huifeng, et al. "DeepFM: a factorization-machine based neural network for CTR prediction." *Proceedings of the 26th International Joint Conference on Artificial Intelligence*. 2017.
[2] Wang, Ruoxi, et al. "Deep & cross network for ad click predictions." *Proceedings of the ADKDD'17*. 2017. 1-7.
[3] Zhang, Weinan, Tianming Du, and Jun Wang. "Deep learning over multi-field categorical data." *European conference on information retrieval*. Springer, Cham, 2016.
[4] Qu, Yanru, et al. "Product-based neural networks for user response prediction." *2016 IEEE 16th International Conference on Data Mining (ICDM)*. IEEE, 2016.
[5] Zhao, Xiangyu, et al. "Autodim: Field-aware embedding dimension searchin recommender systems." *Proceedings of the Web Conference 2021*. 2021.
[6] Wang, Yejing, et al. "Autofield: Automating feature selection in deep recommender systems." *Proceedings of the ACM Web Conference 2022*. 2022.
[7] Shi, Hao-Jun Michael, et al. "Compositional embeddings using complementary partitions for memory-efficient recommendation systems." *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 2020.
[8] Liu, Siyi, et al. "Learnable Embedding sizes for Recommender Systems." *International Conference on Learning Representations*. 2020.

Result

Table 2: Overall Performance Comparison.

	Dataset	DeepFM			DCN			FNN			IPNN		
		AUC	Logloss	Sparsity	AUC	Logloss	Sparsity	AUC	Logloss	Sparsity	AUC	Logloss	Sparsity
Criteo	Original	0.8104	0.4409	-	0.8106	0.4408	-	0.8110	0.4404	-	0.8113	0.4401	-
	AutoDim	0.8093	0.4420	0.8642	0.8096	0.4418	0.7917	0.8104	0.4410	**0.7187**	0.8103	0.4411	**0.7179**
	AutoField	0.8101	0.4412	0.0009	0.8108	0.4405	0.4108	0.8108	0.4406	0.6221	0.8111	0.4403	0.3941
	QR	0.8084	0.4444	0.5000	0.8103	0.4411	0.5000	0.8105	0.4408	0.5000	0.8102	0.4411	0.5000
	PEP	0.7980	0.4541	0.5010	0.8110	0.4404	0.5802	0.8108	0.4406	0.5802	0.8111	0.4402	0.5607
	OptEmbed	**0.8105**	**0.4409**	**0.9684**	**0.8113**	**0.4402**	**0.8534**	**0.8114**	**0.4400**	0.6710	**0.8114**	**0.4401**	0.7122
Avazu	Original	0.7884	0.3751	-	0.7894	0.3748	-	0.7896	0.3748	-	0.7898	0.3745	-
	AutoDim	0.7843	0.3779	**0.6936**	0.7893	0.3744	0.5013	0.7894	**0.3743**	0.5017	0.7894	0.3743	0.3892
	AutoField	0.7866	0.3762	0.0020	0.7887	0.3748	0.0001	0.7892	0.3748	0.0001	0.7897	0.3744	0.0001
	QR	0.7762	0.3821	0.5000	0.7868	0.3766	0.5000	0.7857	0.3769	0.5000	0.7849	0.3781	**0.5000**
	PEP	0.7877	0.3754	0.4126	0.7896	0.3743	0.3016	0.7894	0.3744	0.3016	0.7897	0.3742	0.3016
	OptEmbed	**0.7888***	**0.3750***	0.3927	**0.7901***	**0.3740**	**0.6840**	**0.7902***	0.3744	**0.5563**	**0.7902**	**0.3740***	0.4693
KDD12	Original	0.7962	0.1532	-	0.8010	0.1522	-	0.8008	0.1522	-	0.8007	0.1522	-
	AutoDim	0.7886	0.1550	0.0029	0.8016	0.1520	0.1904	0.8012	0.1522	0.1669	0.8013	0.1521	0.2286
	AutoField	0.7953	0.1534	0.0038	0.8011	0.1525	0.0000	0.8006	0.1522	0.0000	0.8006	0.1522	0.0038
	QR	0.7913	0.1544	0.5000	0.7925	0.1541	**0.5000**	0.7938	0.1538	0.5000	0.7928	0.1540	**0.5000**
	PEP	0.7957	0.1533	0.1001	0.7992	0.1525	0.1003	0.7984	0.1527	0.1003	0.7957	0.1535	0.1003
	OptEmbed	**0.7971***	**0.1530***	**0.6183**	**0.8021***	**0.1519**	0.4715	**0.8027***	**0.1522**	**0.5105**	**0.8028***	**0.1521**	0.4154

Here $*$ denotes statistically significant improvement (measured by a two-sided t-test with p-value < 0.05) over the best baseline.

On Criteo and Avazu, OptEmbed tends to **save model parameters while keeping the performance**.

Result

Table 2: Overall Performance Comparison.

	Dataset	DeepFM			DCN			FNN			IPNN		
		AUC	Logloss	Sparsity	AUC	Logloss	Sparsity	AUC	Logloss	Sparsity	AUC	Logloss	Sparsity
Criteo	Original	0.8104	0.4409	-	0.8106	0.4408	-	0.8110	0.4404	-	0.8113	0.4401	-
	AutoDim	0.8093	0.4420	0.8642	0.8096	0.4418	0.7917	0.8104	0.4410	**0.7187**	0.8103	0.4411	**0.7179**
	AutoField	0.8101	0.4412	0.0009	0.8108	0.4405	0.4108	0.8108	0.4406	0.6221	0.8111	0.4403	0.3941
	QR	0.8084	0.4444	0.5000	0.8103	0.4411	0.5000	0.8105	0.4408	0.5000	0.8102	0.4411	0.5000
	PEP	0.7980	0.4541	0.5010	0.8110	0.4404	0.5802	0.8108	0.4406	0.5802	0.8111	0.4402	0.5607
	OptEmbed	**0.8105**	**0.4409**	**0.9684**	**0.8113**	**0.4402**	**0.8534**	**0.8114**	**0.4400**	0.6710	**0.8114**	**0.4401**	0.7122
Avazu	Original	0.7884	0.3751	-	0.7894	0.3748	-	0.7896	0.3748	-	0.7898	0.3745	-
	AutoDim	0.7843	0.3779	**0.6936**	0.7893	0.3744	0.5013	0.7894	**0.3743**	0.5017	0.7894	0.3743	0.3892
	AutoField	0.7866	0.3762	0.0020	0.7887	0.3748	0.0001	0.7892	0.3748	0.0001	0.7897	0.3744	0.0001
	QR	0.7762	0.3821	0.5000	0.7868	0.3766	0.5000	0.7857	0.3769	0.5000	0.7849	0.3781	**0.5000**
	PEP	0.7877	0.3754	0.4126	0.7896	0.3743	0.3016	0.7894	0.3744	0.3016	0.7897	0.3742	0.3016
	OptEmbed	**0.7888***	**0.3750***	0.3927	**0.7901***	**0.3740**	**0.6840**	**0.7902***	0.3744	**0.5563**	**0.7902**	**0.3740***	0.4693
KDD12	Original	0.7962	0.1532	-	0.8010	0.1522	-	0.8008	0.1522	-	0.8007	0.1522	-
	AutoDim	0.7886	0.1550	0.0029	0.8016	0.1520	0.1904	0.8012	0.1522	0.1669	0.8013	0.1521	0.2286
	AutoField	0.7953	0.1534	0.0038	0.8011	0.1525	0.0000	0.8006	0.1522	0.0000	0.8006	0.1522	0.0038
	QR	0.7913	0.1544	0.5000	0.7925	0.1541	**0.5000**	0.7938	0.1538	0.5000	0.7928	0.1540	**0.5000**
	PEP	0.7957	0.1533	0.1001	0.7992	0.1525	0.1003	0.7984	0.1527	0.1003	0.7957	0.1535	0.1003
	OptEmbed	**0.7971***	**0.1530***	**0.6183**	**0.8021***	**0.1519**	0.4715	**0.8027***	**0.1522**	**0.5105**	**0.8028***	**0.1521**	0.4154

Here * denotes statistically significant improvement (measured by a two-sided t-test with p-value < 0.05) over the best baseline.

On KDD12, OptEmbed tends to **boost model performance**.

Ablation

Table 3: Performance Comparison for Component Analysis.

	Basic Model	Metrics	AUC	Logloss	Sparsity
Criteo	DeepFM	Original	0.8104	0.4409	-
		OptEmbed-E	0.8104	0.4410	0.6267
		OptEmbed-D	0.8103	0.4410	0.5547
		OptEmbed	0.8105	0.4409	0.9684
	DCN	Original	0.8106	0.4408	-
		OptEmbed-E	0.8110	0.4404	0.6111
		OptEmbed-D	0.8110	0.4403	0.7192
		OptEmbed	0.8113	0.4402	0.8534
Avazu	DeepFM	Original	0.7884	0.3751	-
		OptEmbed-E	0.7884	0.3752	0.0000
		OptEmbed-D	0.7888	0.3750	0.3927
		OptEmbed	0.7888	0.3750	0.3927
	DCN	Original	0.7894	0.3748	-
		OptEmbed-E	0.7895	0.3746	0.0024
		OptEmbed-D	0.7900	0.3740	0.5044
		OptEmbed	0.7900	0.3743	0.6840

OptEmbed-E: only using embedding mask
OptEmbed-D: only using field-wise dimension mask

Table 4: Ablation About Re-training Stage.

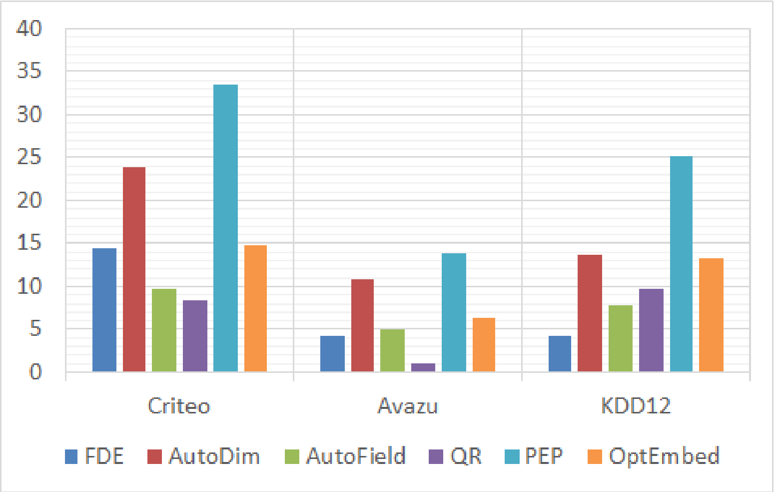
Dataset	Criteo		Avazu		KDD12	
Retrain	w.	w.o.	w.	w.o.	w.	w.o.
AUC	0.8113	0.8110	0.7900	0.7895	0.8021	0.8005
Logloss	0.4402	0.4404	0.3743	0.3749	0.1523	0.1526

w. stands for with re-training. *w.o.* stands for without re-training.

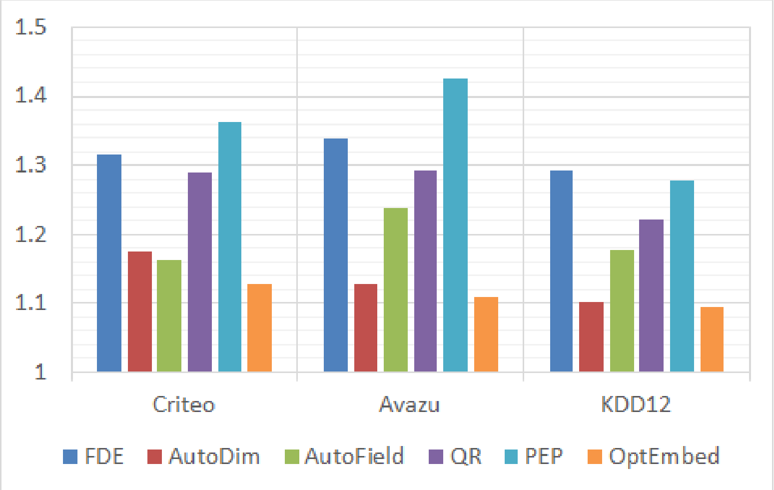
- Retraining is necessary.

- On Criteo, both components reduce the embedding parameters.
- On Avazu, OptEmbed-E makes no significant difference compared to original model.

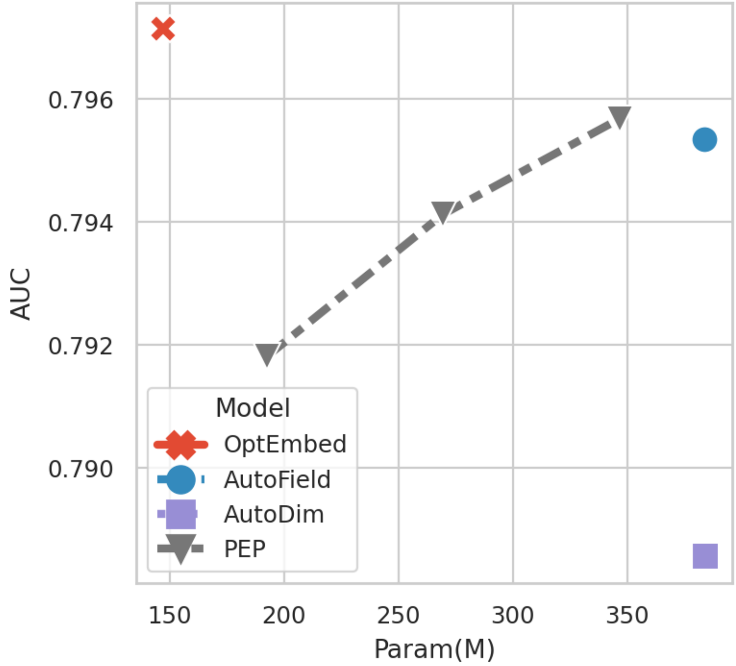
Efficiency Analysis



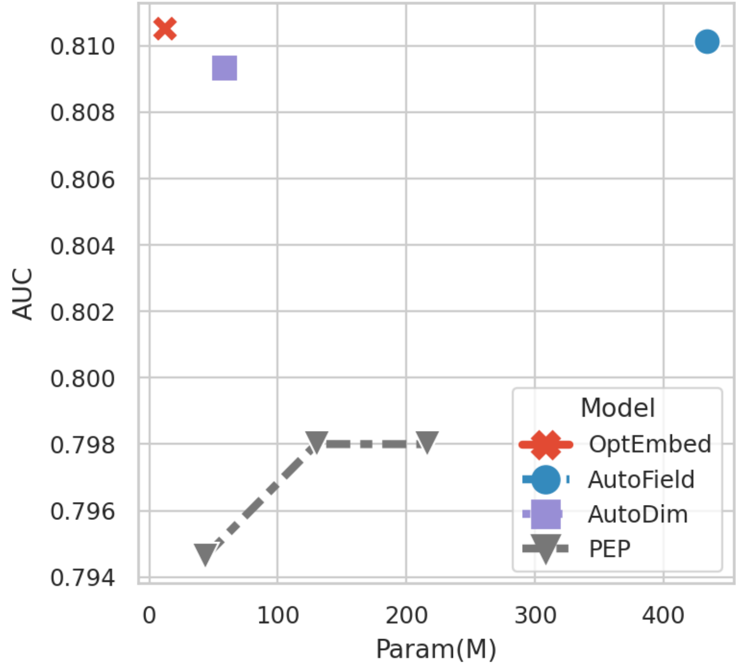
(a) Training Time (h)



(b) Inference Time (ms)



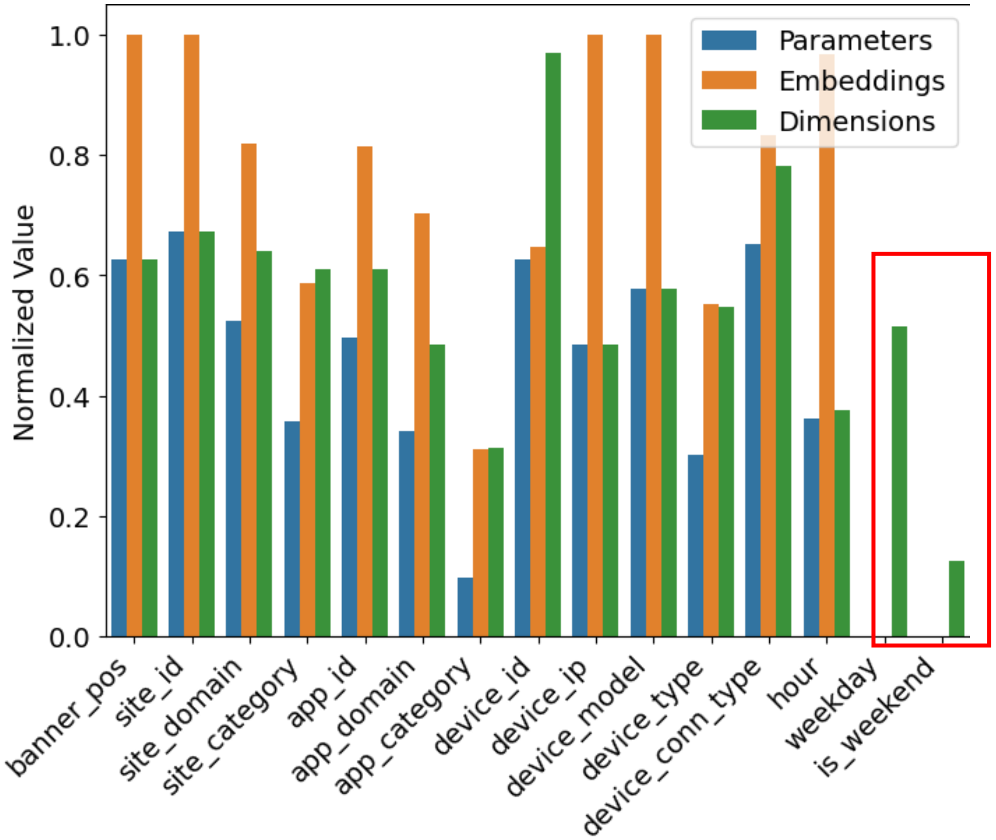
(a) Criteo



(b) Avazu

- OptEmbed tends to perform best in terms of **Param-AUC tradeoff**.
- OptEmbed ranks medium-level for training time and performs best for **inference time**.

Case Study



- Perform on Avazu dataset with DeepFM model.
- Filter out unnecessary fields completely introduced by **the common best practice**.

Conclusion

1. We first propose three requirements for an **optimal embedding table: N**o **R**edundant **F**eature, **E**mbedding **D**imension **F**lexible and **H**ardware **F**riendly.
2. Based on these requirements, **a novel, model-agnostic framework OptEmbed** is proposed, which optimizes the embedding table in a unifying way.
3. Extensive experiments demonstrate the superiority of OptEmbed in model performance, runtime efficiency and model size reduction.

Thanks for Listening!

1. We first propose three requirements for an **optimal embedding table:** **N**o **R**edundant **F**eature, **E**mbedding **D**imension **F**lexible and **H**ardware **F**riendly.
2. Based on these requirements, **a novel, model-agnostic framework OptEmbed** is proposed, which optimizes the embedding table in a unifying way.
3. Extensive experiments demonstrate the superiority of OptEmbed in model performance, runtime efficiency and model size reduction.

For code implementation, kindly check:
https://github.com/fuyuanlyu/OptEmbed

Formulation

For n field one-hot encoded raw input:

$$x = [x_{(1)}, x_{(2)}, \ldots, x_{(n)}]$$

Transform into dense vector:

$$e_{(i)} = E \times x_{(i)}$$
$$e = [e_{(1)}, e_{(2)}, \ldots, e_{(n)}]$$

Fed into feature interaction and classification layer:

$$\hat{y} = \mathcal{F}(\mathrm{E} \times x|\mathrm{W})$$

Adopt the cross-entropy loss:

$$\mathrm{CE}(y, \hat{y}) = y \log(\hat{y}) + (1 - y) \log(1 - \hat{y}).$$

Formulate the CTR prediction problem:

$$\min_{\mathrm{E},\mathrm{W}} \mathcal{L}_{\mathrm{CE}}(\mathcal{D}|\{\mathrm{E}, \mathrm{W}\}) = -\frac{1}{|\mathcal{D}|} \sum_{(\mathbf{x}, y) \in \mathcal{D}} \mathrm{CE}(y, \mathcal{F}(\mathrm{E} \times \mathbf{x}|\mathrm{W}))$$

Fulfill the three requirements, we decompose the original single embedding table into a series of field-wise embedding table:

$$E = [E_{(1)}, E_{(2)}, \ldots, E_{(n)}], E_{(i)} \in \mathrm{R}^{|\mathrm{f_i}| \times D_i}$$

For **R1**: **N**o **R**edundant **F**eatures in $|f|$

$$\sum_{i=1}^{n} |f_{(i)}| \leq |f|$$

R2 and **R3** are naturally satisfied.

Re-formulate the CTR prediction problem:

$$\sum_{i=1}^{n} |f_{(i)}| \leq |f|$$

Optimal Embedding Table

Fulfill the three requirements, we decompose the original single embedding table into a series of field-wise embedding table:

$$E = \left[E_{(1)}, E_{(2)}, \ldots, E_{(n)}\right], E_{(i)} \in \mathrm{R}^{|\mathrm{f}_{(\mathrm{i})}| \times D_{(i)}}$$

For **R1**: **N**o **R**edundant **F**eatures:

$$\sum_{i=1}^{n} |f_{(i)}| \leq |f|$$

R2: **E**mbedding **D**imension **F**lexible and **R3**: **H**ardware **F**riendly are naturally satisfied.

Re-formulate the CTR prediction problem:

$$\min_{\mathrm{E}^*, \mathrm{W}} \mathcal{L}_{\mathrm{CE}}(\mathcal{D} | \{\mathrm{E}^*, \mathrm{W}\}), \ \mathrm{E}^* = [\mathrm{E}_{(1)}, \mathrm{E}_{(2)}, \cdots, \mathrm{E}_{(n)}],$$

$$s.t. \ \mathrm{E}_{(i)} \in \mathbb{R}^{|f_{(i)}| \times D_{(i)}}, \ \sum_{i=1}^{n} |f_{(i)}| \leq |f|, \ D_{(i)} \leq D, \ \forall i \leq n.$$

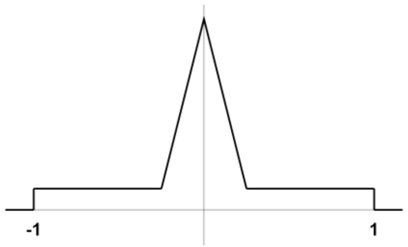
Redundant Embedding Pruning

Inspired by **network pruning** [1,2], we re-parameterize embedding mask:

$$m_e = S(L_\beta(E) - t) \in \{0,1\}^{|f|}$$

$t \in R^{|n|}$ is the field-wise threshold vector, L_β indicates the L_β norm of each feature embedding, $S()$ is the unit step function with long-tail estimator [1].

$$S(x) = \begin{cases} 1, x > 0 \\ 0, x \leq 0 \end{cases} \qquad \frac{d}{dx}S(\mathrm{x}) \approx \mathrm{H}(x) = \begin{cases} 2 - 4|x|, |x| \leq 0.4 \\ 0.4, 0.4 < |x| \leq 1 \\ 0, |x| > 1 \end{cases}$$



[1] Junjie, L. I. U., et al. "Dynamic Sparse Training: Find Efficient Sparse Network From Scratch With Trainable Masked Layers." *International Conference on Learning Representations*. 2019.
[2] Yuan, Xin, Pedro Henrique Pamplona Savarese, and Michael Maire. "Growing Efficient Deep Networks by Structured Continuous Sparsification." *International Conference on Learning Representations*. 2020.

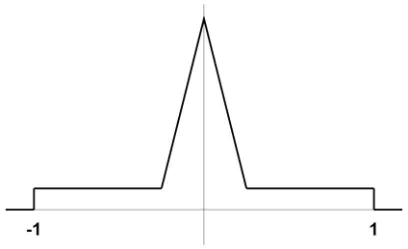
Redundant Embedding Pruning

Inspired by **network pruning** [1,2], we re-parameterize embedding mask:

$$m_e = S(L_\beta(E) - t) \in \{0,1\}^{|f|}$$

$t \in R^{|n|}$ is the field-wise threshold vector, L_β indicates the L_β norm of each feature embedding, $S()$ is the unit step function with long-tail estimator [1].

$$S(x) = \begin{cases} 1, x > 0 \\ 0, x \leq 0 \end{cases} \qquad \frac{d}{dx} S(\mathrm{x}) \approx \mathrm{H}(x) = \begin{cases} 2 - 4|x|, |x| \leq 0.4 \\ 0.4, 0.4 < |x| \leq 1 \\ 0, |x| > 1 \end{cases}$$



Sparse regularization Term: $L_s = \sum_{i=1}^{n} \exp(-t_i)$

The final objective in this stage:

$$\min_{\mathbf{m}_e, \mathbf{E}, \mathbf{W}} \mathcal{L}_{\mathrm{CE}}(\mathcal{D} | \{\hat{\mathbf{E}}, \mathbf{W}\}) + \alpha \mathcal{L}_s, \ \hat{\mathbf{E}} = \mathbf{E} \odot \mathbf{m}_e.$$

[1] Junjie, L. I. U., et al. "Dynamic Sparse Training: Find Efficient Sparse Network From Scratch With Trainable Masked Layers." *International Conference on Learning Representations*. 2019.
[2] Yuan, Xin, Pedro Henrique Pamplona Savarese, and Michael Maire. "Growing Efficient Deep Networks by Structured Continuous Sparsification." *International Conference on Learning Representations*. 2020.

Embedding Dimension Search

Formulated into a **one-shot neural architecture search** problem [1,2]:

$$\mathbf{m}_d^* = \underset{\mathbf{m}_d \in \mathcal{S}_e}{\arg\min}\, \mathcal{L}_{\mathrm{CE}}(\mathcal{D}_{val} | \{\hat{\mathbf{E}}_s \odot \mathbf{m}_d, \hat{\mathbf{W}}_s\}),$$

$$s.t.\ \{\hat{\mathbf{E}}_s, \hat{\mathbf{W}}_s\} = \underset{\{\mathbf{E}_s, \mathbf{W}_s\} \in \Omega}{\arg\min}\ \mathbb{E}_{\mathbf{m}_d \sim \Gamma(\mathcal{S}_e)} \mathcal{L}_{\mathrm{CE}}(\mathcal{D} | \{\mathbf{E}_s \odot \mathbf{m}_d, \mathbf{W}_s\}),$$

supernet $\{E_s^{\wedge}, W_s^{\wedge}\}$

where $S_e = \{1, 2, \dots, D\}$ denotes the search space, $\Gamma(S_e)$ is the prior distribution of the search space (uniform distribution in this case).

[1] Bender, Gabriel, et al. "Understanding and simplifying one-shot architecture search." *International conference on machine learning*. PMLR, 2018.
[2] Guo, Zichao, et al. "Single path one-shot neural architecture search with uniform sampling." *European conference on computer vision*. Springer, Cham, 2020.

Embedding Dimension Search

Formulated into a **one-shot neural architecture search** problem [1,2]:

$$\mathbf{m}_d^* = \underset{\mathbf{m}_d \in \mathcal{S}_e}{\arg\min}\, \mathcal{L}_{\mathrm{CE}}(\mathcal{D}_{val} | \{\hat{\mathbf{E}}_s \odot \mathbf{m}_d, \hat{\mathbf{W}}_s\}),$$

$$s.t.\ \{\hat{\mathbf{E}}_s, \hat{\mathbf{W}}_s\} = \underset{\{\mathbf{E}_s, \mathbf{W}_s\} \in \Omega}{\arg\min}\ \mathbb{E}_{\mathbf{m}_d \sim \Gamma(\mathcal{S}_e)} \mathcal{L}_{\mathrm{CE}}(\mathcal{D} | \{\mathbf{E}_s \odot \mathbf{m}_d, \mathbf{W}_s\}),$$

supernet $\{E_s^{\wedge}, W_s^{\wedge}\}$

where $S_e = \{1, 2, \dots, D\}$ denotes the search space, $\Gamma(S_e)$ is the prior distribution of the search space (uniform distribution in this case).

The supernet training **is aligned with** the previous redundant embedding pruning:

$$\min_{\mathbf{m}_e, \mathbf{E}, \mathbf{W}} \mathbb{E}_{\mathbf{m}_d \sim \mathrm{Uniform}(\mathcal{S}_e)} \mathcal{L}_{\mathrm{CE}}(\mathcal{D} | \{\hat{\mathbf{E}}, \mathbf{W}\}) + \alpha \mathcal{L}_s,$$

$$\hat{\mathbf{E}} = \mathbf{E}_s \odot \mathbf{m}_d = \mathbf{E} \odot \mathbf{m}_e \odot \mathbf{m}_d.$$

After obtaining the supernet $\{E_s^{\wedge}, W_s^{\wedge}\}$, we adopt evolutional search to find dimension mask m_d^*.

[1] Bender, Gabriel, et al. "Understanding and simplifying one-shot architecture search." *International conference on machine learning*. PMLR, 2018.
[2] Guo, Zichao, et al. "Single path one-shot neural architecture search with uniform sampling." *European conference on computer vision*. Springer, Cham, 2020.

Re-training

Re-train the embedding E and the model weights W given:

- the embedding mask m_e^*
- field-wise dimension mask m_d^*.

$$\mathrm{argmin}_{\mathbf{E},\mathbf{W}} \mathcal{L}_{CE}(\mathcal{D}|\{\mathbf{E} \odot \mathbf{m}_e^* \odot \mathbf{m}_d^*, \mathbf{W}\}).$$

Algorithm 1 The OptEmbed Algorithm

Require: training dataset $\mathcal{D}$, validation dataset $\mathcal{D}_{val}$
Ensure: optimal embedding table $\mathbf{E}^*$ and model parameters $\mathbf{W}^*$

Supernet Training and Embedding Pruning
while not converge **do**
    Sample a mini-batch from the training dataset
    $\{\hat{\mathbf{E}}_s, \hat{\mathbf{W}}_s\}, \mathbf{m}_e$ = SupernetTrain($\mathcal{D}$) ▷ Eq. 16
end while
$\mathbf{m}_e^*$ = GetBestPerform($\{\mathbf{m}_e\}$)
Dimension Mask Searching
$\tau = 0$; P_τ = Initialize_population($n_m + n_c$); Topk = $\emptyset$;
while $\tau < T$ **do**
    AUC$_\tau$ = Inference($\hat{\mathbf{E}}_s, \hat{\mathbf{W}}_s, \mathcal{D}_{val}, P_\tau$);
    Topk = Update_Topk(Topk, P_τ, AUC$_\tau$);
    P_τ^c = Crossover(Topk, n_c);
    P_τ^m = Mutation(Topk, n_m, $prob$);
    $P_{\tau+1} = P_\tau^m \cup P_\tau^c$;
    $\tau = \tau + 1$;
end while
$\mathbf{m}_d^*$ = GetBestCand(P_τ) ▷ Eq. 15
Re-training##
Retrain $\{\mathbf{E}^*, \mathbf{W}^*\}$ given $\mathbf{m}_e^*$ and $\mathbf{m}_d^*$ ▷ Eq. 17