# Tabular Deep Learning

Lecturer: Artem Babenko

Research

ASCOMP 2024

# Lecturer

- Artem Babenko, Research Lead @ Yandex Research
- Publications on deep/machine learning for tabular data by Yandex Research
  - (NeurIPS 2018) CatBoost: unbiased boosting with categorical features
  - (ICLR 2020) Neural Oblivious Decision Ensembles for Deep Learning on Tabular Data
  - (NeurIPS 2021) Revisiting Deep Learning Models for Tabular Data
  - (NeurIPS 2022) On Embeddings for Numerical Features in Tabular Deep Learning
  - (arXiv 2022) Revisiting Pretraining Objectives for Tabular Deep Learning
  - (ICML 2023) TabDDPM: Modelling Tabular Data with Diffusion Models
  - (ICLR 2024) TabR: Tabular Deep Learning Meets Nearest Neighbors
  - (2024) Several projects under submission
- Tabular DL projects by Yandex Research: github.com/yandex-research/rtdl (RTDL = Research on Tabular Deep Learning)

# YR Tabular DL team

Artem Babenko

Yura Gorishniy

Nikolay Kartashev

Akim Kotelnikov

Ivan Rubachev

# Outline

- Introduction
- The pre-deep learning era of Tabular ML
- Modern Tabular Deep Learning
- Real-world impact

# Outline

- Introduction
- The pre-deep learning era of Tabular ML
- Modern Tabular Deep Learning
- Real-world impact

# Tabular data

Tabular data — two-dimensional tables
- rows ~ objects
- columns ~ features

Today we focus on
- supervised regression
- supervised classification

Applications
- everyday tasks…
- …and many others

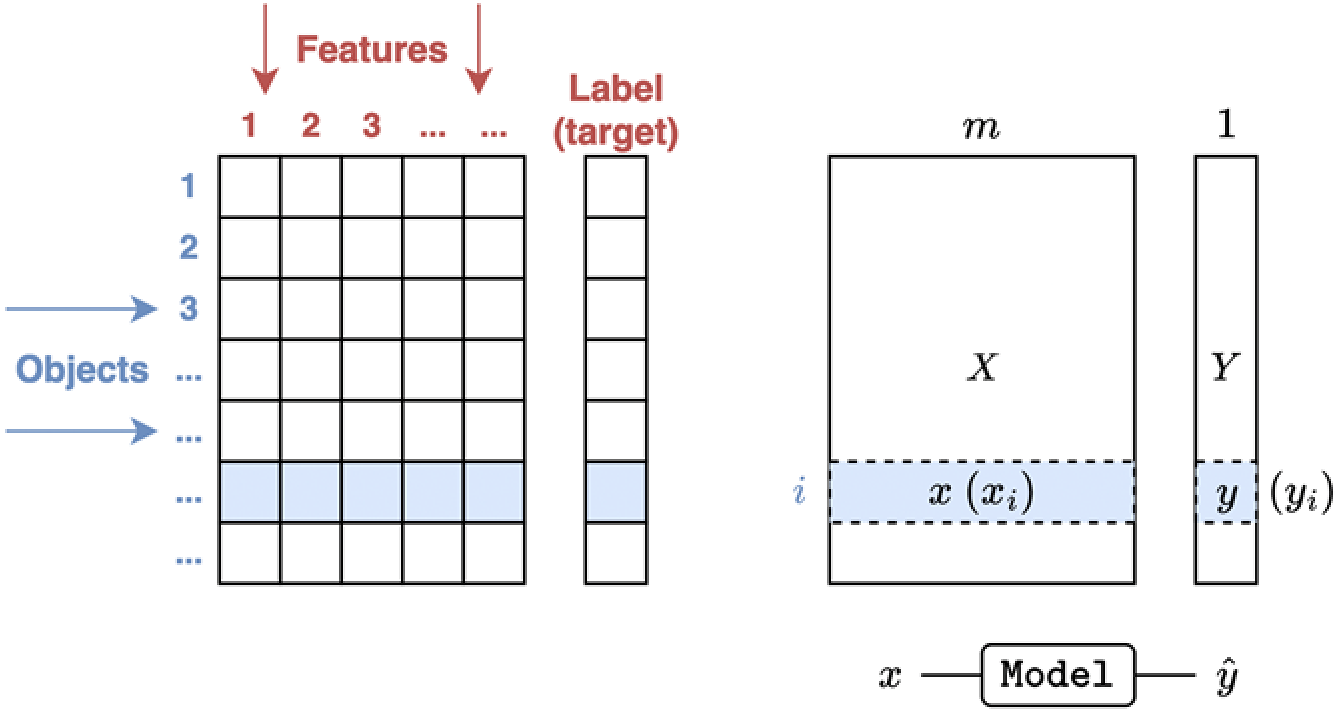
| A | B |
|---|---|
| ... | ... |
| ... | ... |
| ... | ... |

X

| C |
|---|
| ... |
| ... |
| ... |

y

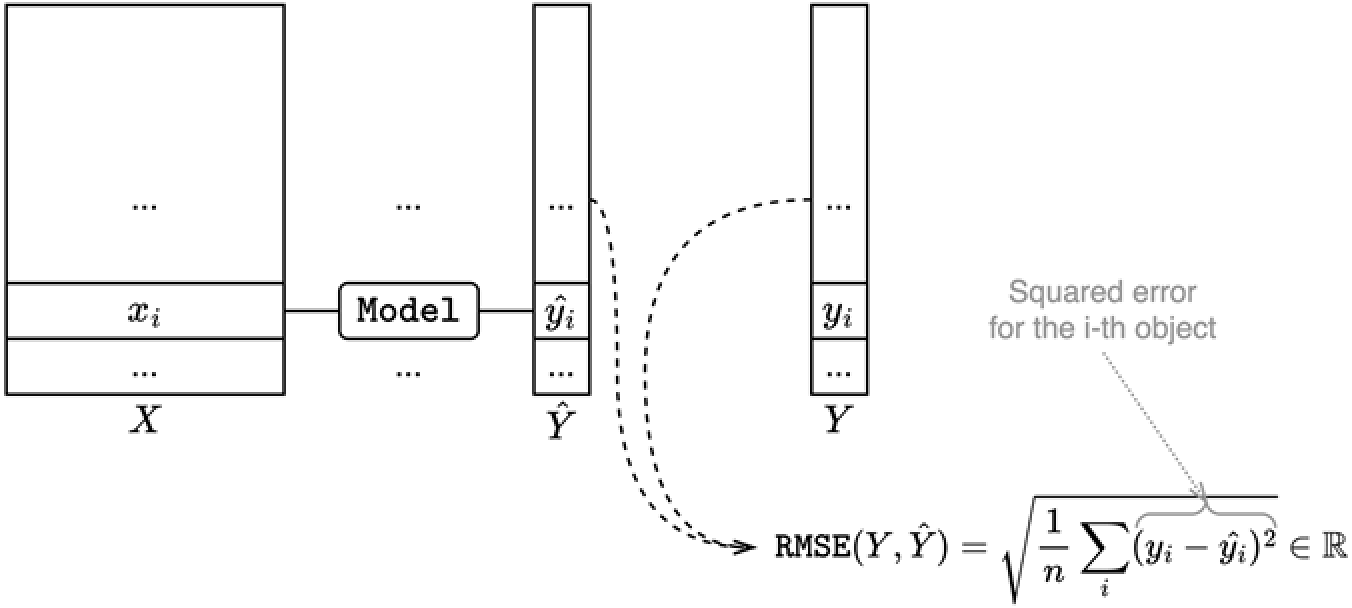
# Notation



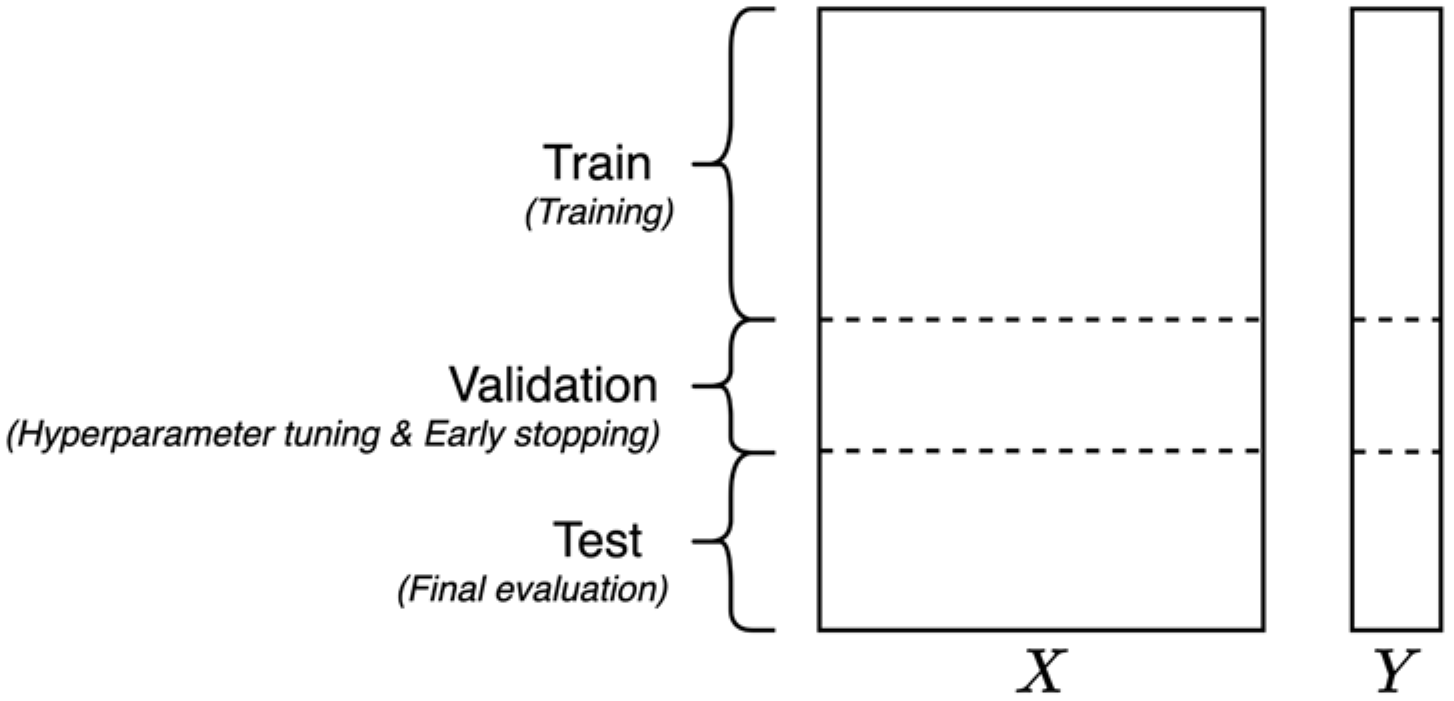
Features
Label
(target)
1 2 3 ... ...
Objects
1
2
3
...
...
...
...
m
1
X
Y
i
x (x_i)
y
(y_i)
x
Model
ŷ

# Metrics

Metrics are used to evaluate how well predictions approximate labels.
Example: Root Mean Squared Error (RMSE)



# Dataset splitting



# Data preprocessing

Continuous features

- QuantileTransformer
- QuantileTransformer with noise (example)
- StandardScaler
- Missing data: x → (0, 1) if x is NaN else (x, 0)

Categorical features

- One-hot encoding
  (typically used when the number of distinct values is not too high)
- Embeddings
- Missing data: make NaN a new category

Binary features

- Just encode as {0, 1}
- Missing data: any reasonable strategy (see “Continuous” and “Categorical”)

Ordinal features

- OrdinalEncoder
- Thermometer encoding
- Cumulative embeddings

**P.S. Standardize regression labels**

# Specifics of Tabular ML problems

- Limited dataset sizes
- Heterogeneous and mixed-type features
- Each problem has its own nature
- Target dependencies are often “ill-behaved”

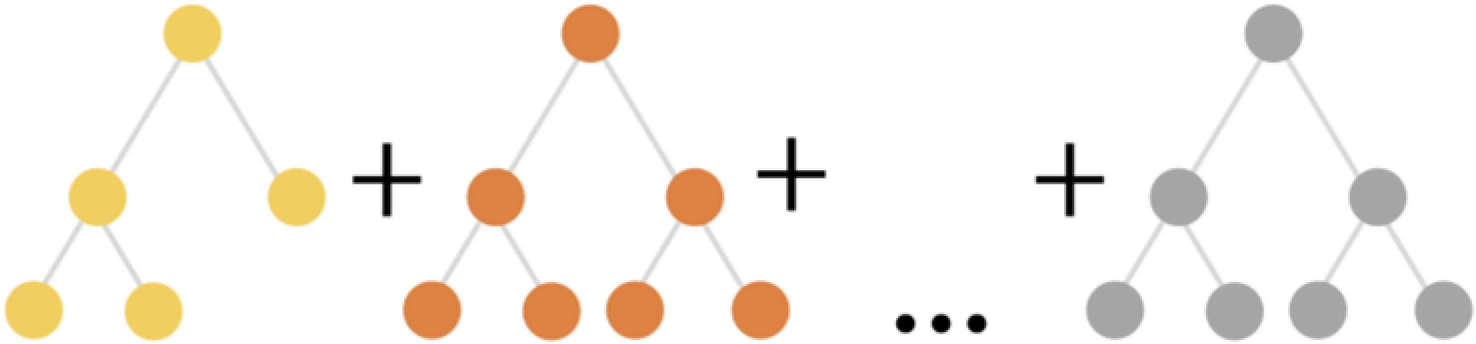
# Outline

- Introduction
- The pre-deep learning era of Tabular ML
- Modern Tabular Deep Learning
- Real-world impact

# Classic machine learning algorithms

- K-Nearest neighbors
- Linear model (Linear regression, Logistic regression, …)
- Support vector machine (SVM)
- Decision tree
- Random forest
- Gradient-boosted decision tree (GBDT)

Gradient Boosting Decision Trees (GBDT)



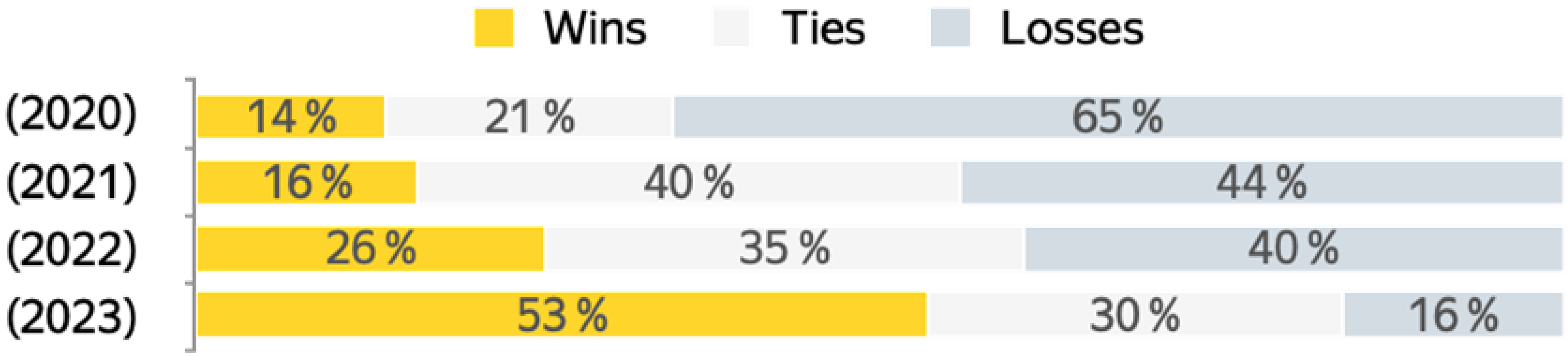
dmlc XGBoost

Yandex CatBoost

LightGBM

# GBDT is a strong baseline for Tabular ML

- Efficient
- Easy-to-use
- Effective




Best DL model vs XGBoost on the academic benchmark of ~40 datasets

# Outline

- Introduction
- The pre-deep learning era of Tabular ML
- Modern Tabular Deep Learning
- Real-world impact

# Chaos in Tabular DL before 2021

Differentiable trees

- NODE (Popov et al., 2020)

“Attention”-based models

- AutoInt (Song et al., 2019)
- TabNet (Arik and Pfister, 2020)

Multiplicative feature interactions

- DCN2 (Wang et al., 2020)

Specific activation functions

- SNN (Klambauer et al., 2017)

Boosting-like models

- GrowNet (Badirli et al., 2020)

And many others

- …

# 2021: Are we really making progress in Tabular DL? [1,2,3]

- Tuning protocols and evaluation are often unfair
- GBDT is still superior to DL
- Sophisticated DL models are often inferior to simple ones

[1] Revisiting Deep Learning Models for Tabular Data, Gorishniy et al., 2021

[2] Tabular Data: Deep Learning is not all you need, Schwartz-Ziv et al., 2021

[3] Regularization is all you need: simple neural nets can excel on tabular data, Kadra et al., 2021

# Outline

- Introduction
- The pre-deep learning era of Tabular ML
- Modern Tabular Deep Learning
    - MLP, Resnet, FT-Transformer
- Real-world impact

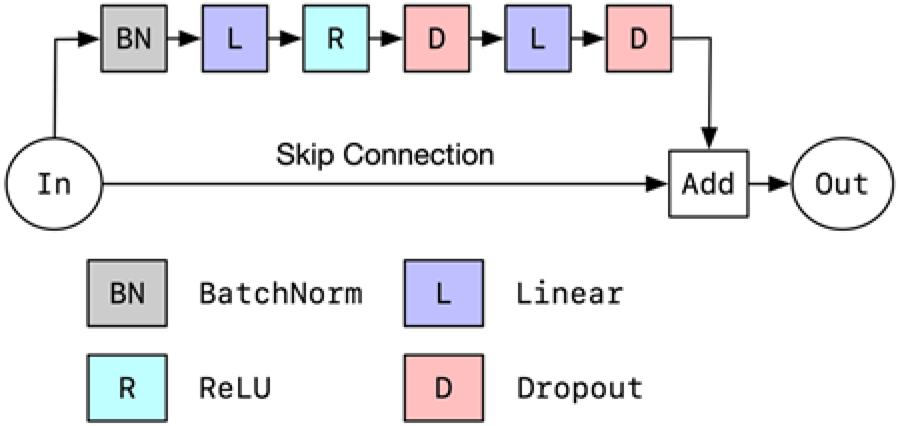
# MLP

- Simple and fast
- Average performance


*One MLP block*

# ResNet for Tabular Data

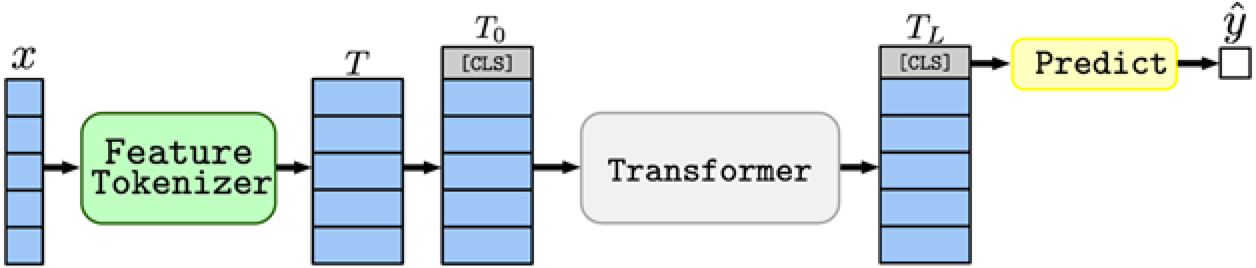
- Inspired by ResNet (He at al., 2015)
- Quite simple and relatively fast
- Hopefully, more powerful than MLP




*One ResNet block*

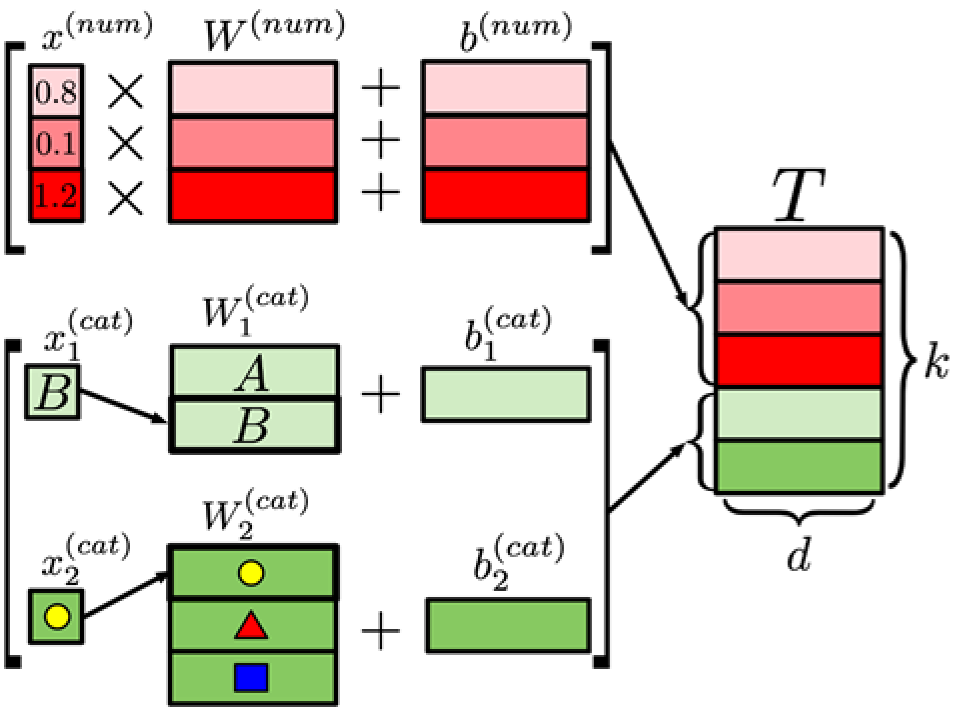
# FT-Transformer (Ours)

- Based on Transformer (Vaswani et al., 2017)
- Slower than ResNet
- Hopefully, more powerful than MLP and ResNet

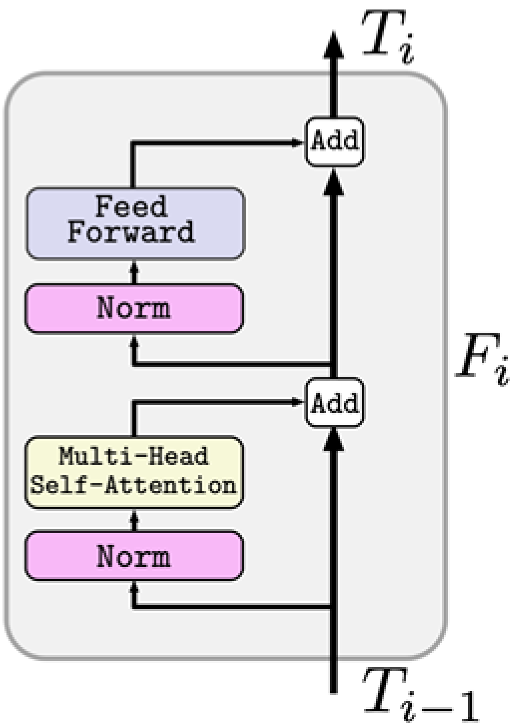



*FT-Transformer*

# FT-Transformer (Ours)




*Feature Tokenizer*




*One Transformer block*

# Experiments

# Experiments: datasets and protocol

| Dataset | N | K | Metric |
|---|---|---|---|
| California Housing | 21K | 8 | RMSE |
| Adult | 49K | 14 | Accuracy (B) |
| Helena | 66K | 27 | Accuracy (M) |
| Jannis | 84K | 54 | Accuracy (M) |
| Higgs (small) | 99K | 28 | Accuracy (B) |
| ALOI | 108K | 128 | Accuracy (M) |
| Epsilon | 500K | 2000 | Accuracy (B) |
| Year | 516K | 90 | RMSE |
| Covtype | 582K | 54 | Accuracy (M) |
| Yahoo | 710K | 699 | RMSE |
| Microsoft | 1201K | 136 | RMSE |

*N ~ dataset size*
*K ~ number of features*

*B ~ binary*
*M ~ multiclass*

- Tuning
  - mostly Optuna (Akiba et al., 2019) (50-100 iterations)
  - grid search from original papers
- Evaluation
  - 15 random seeds
  - ensembles: three ensembles (each consists of five single models)
- No DL tricks
  - no augmentation
  - no lr scheduling
  - no pretraining
  - etc.

# Experiments: Neural Networks

| Model | Average rank (std) |
|---|---|
| **TabNet** | 7.5 (2.0) |
| **SNN** | 6.4 (1.4) |
| **AutoInt** | 5.7 (2.3) |
| **GrowNet** | 5.7 (2.2) |
| **MLP** | 4.8 (1.9) |
| **DCN V2** | 4.7 (2.0) |
| **NODE** | 3.9 (2.8) |
| **ResNet** | 3.3 (1.8) |
| **FT-Transformer** | 1.8 (1.2) |

**Takeaways**

- MLP is still a good sanity check
- ResNet is a strong baseline
- FT-Transformer outperforms existing solutions on most of the tasks
- Tuning matters

# Experiments: FT-Transformer vs GBDT (ensembles)

| *Dataset* | **CA** ↓ | **AD** ↑ | **HE** ↑ | **JA** ↑ | **HI** ↑ | **AL** ↑ | **EP** ↑ | **YE** ↓ | **CO** ↑ | **YA** ↓ | **MI** ↓ |
|---|---|---|---|---|---|---|---|---|---|---|---|
| *#objects* | *20K* | *49K* | *65K* | *84K* | *98K* | *108K* | *500K* | *515K* | *581K* | *710K* | *1200K* |
| **XGBoost (d)** | 0.462 | **0.874** | 0.348 | 0.711 | 0.717 | 0.924 | 0.88 | 9.192 | 0.964 | 0.761 | 0.751 |
| **CatBoost (d)** | **0.428** | 0.873 | 0.386 | 0.724 | 0.728 | 0.948 | 0.889 | 8.885 | 0.91 | 0.749 | 0.744 |
| **FT-Transformer (d)** | 0.454 | 0.86 | **0.395** | **0.734** | **0.731** | **0.966** | **0.897** | **8.727** | **0.973** | **0.747** | **0.742** |
| **FT-Transformer*** | 0.448 | 0.86 | 0.398 | 0.739 | 0.731 | 0.967 | 0.898 | 8.751 | 0.973 | 0.747 | 0.743 |

*(d) ~ default configuration* **out of competition* ↑ *Accuracy* ↓ *RMSE* **Best**

**Takeaways**

- ensemble of default FT-Transformers is a powerful thing

# Experiments: ResNet & FT-Transformer vs GBDT (ensembles)

| *Dataset* | CA ⬇ | AD ⬆ | HE ⬆ | JA ⬆ | HI ⬆ | AL ⬆ | EP ⬆ | YE ⬇ | CO ⬆ | YA ⬇ | MI ⬇ |
|---|---|---|---|---|---|---|---|---|---|---|---|
| *#objects* | *20K* | *49K* | *65K* | *84K* | *98K* | *108K* | *500K* | *515K* | *581K* | *710K* | *1200K* |
| **XGBoost** | 0.431 | 0.872 | 0.377 | 0.724 | 0.728 | - | 0.886 | 8.819 | 0.969 | **0.732** | 0.742 |
| **CatBoost** | **0.423** | **0.874** | 0.388 | 0.727 | 0.729 | - | 0.89 | 8.837 | 0.968 | 0.74 | **0.741** |
| **ResNet** | 0.478 | 0.857 | **0.398** | 0.734 | **0.731** | 0.966 | **0.898** | 8.77 | 0.967 | 0.751 | 0.745 |
| **FT-Transformer** | 0.448 | 0.86 | **0.398** | **0.739** | **0.731** | **0.967** | **0.898** | **8.751** | **0.973** | 0.747 | 0.743 |

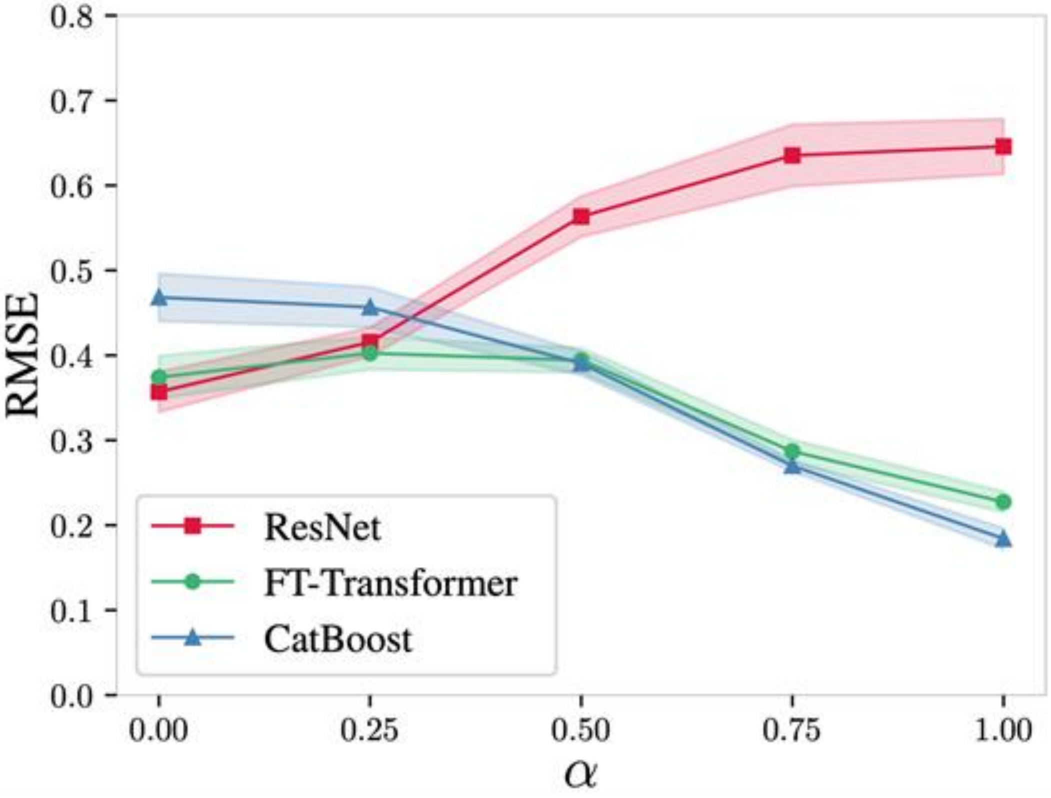
⬆ Accuracy ⬇ RMSE

**Best**

**Takeaways**

- “DL vs GBDT” is an open problem
- **FT-Transformer reduces the gap between ResNet and GBDT**

# An intriguing property of FT-Transformer




$$x \sim \mathcal{N}(0, I_k),$$
$$y = \alpha \cdot f_{GBDT}(x) + (1 - \alpha) \cdot f_{DNN}(x).$$

$f_{GBDT}$ ~ easy for GBDT

$f_{DNN}$ ~ easy for ResNet

**Takeaways**

- FT-Transformer is a more universal architecture for Tabular Data
- Further research is needed to understand this phenomenon

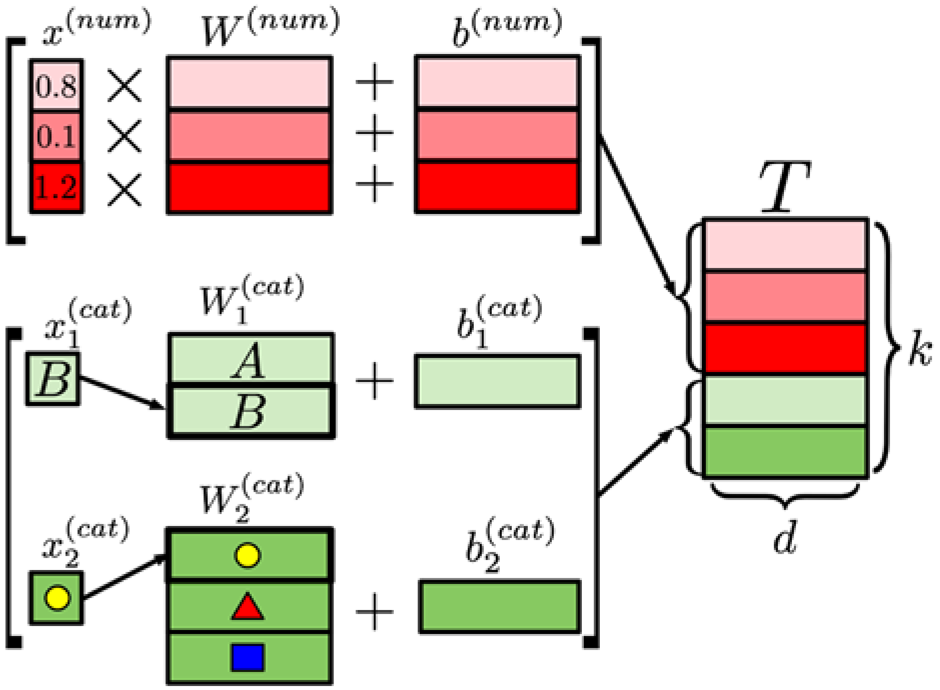
# Conclusion

- **MLP and ResNet**
  - fast and strong baselines
- **FT-Transformer**
  - slower
  - can yield even better performance
- FT-Transformer is a more universal architecture for Tabular Data
- Python package with the new models:
  `pip install rtdl`
- Source code:
  https://github.com/yandex-research/rtdl

# Outline

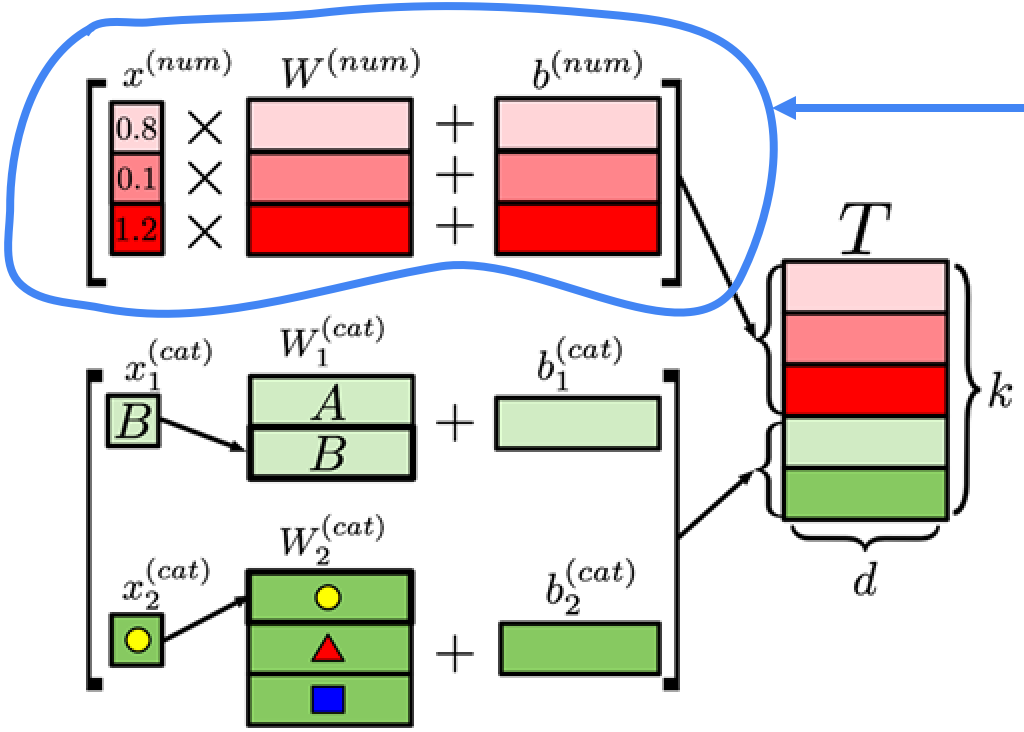
- Introduction
- The pre-deep learning era of Tabular ML
- Modern Tabular Deep Learning
    - Embeddings for Numerical Features
- Real-world impact

# How can we improve FT-Transformer?




*(2022) On Embeddings for Numerical Features in Tabular Deep Learning*

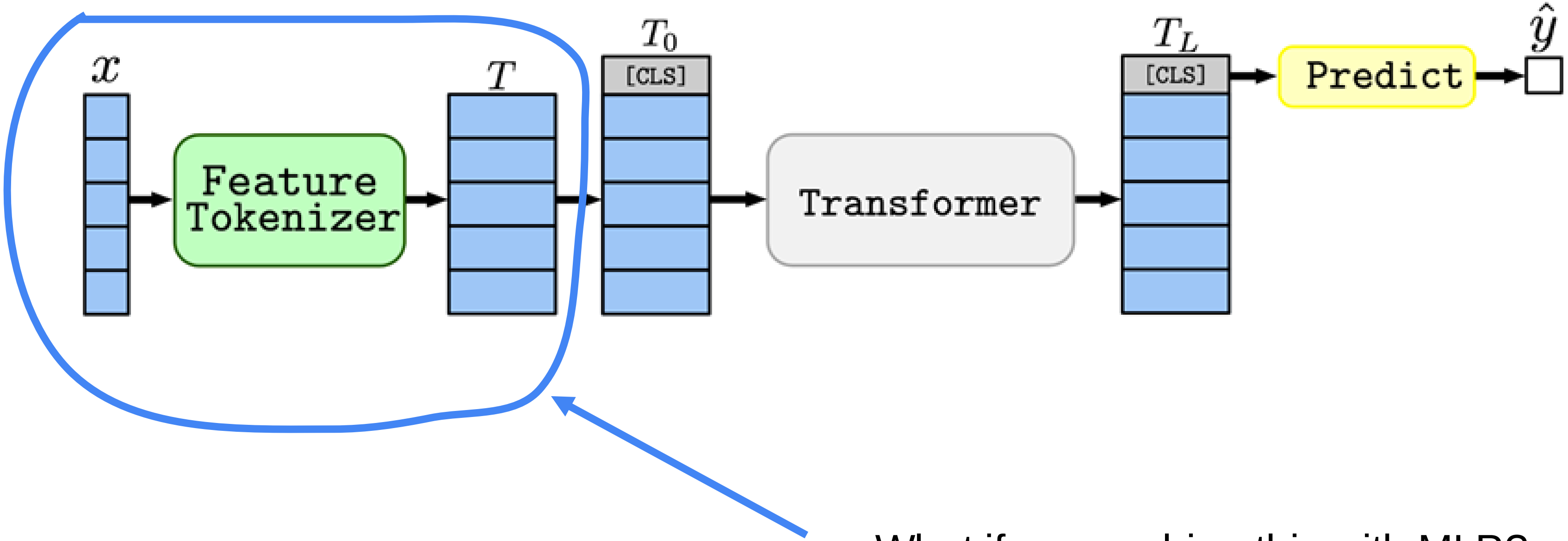
# How can we improve FT-Transformer?




Looks too simple

*(2022) On Embeddings for Numerical Features in Tabular Deep Learning*

# But wait…




What if we combine this with MLP?

*(2022) On Embeddings for Numerical Features in Tabular Deep Learning*

# Moreover…

- Transformers perform well
  - The only model with embeddings for **numerical features**

*(2022) On Embeddings for Numerical Features in Tabular Deep Learning*

# Moreover…

- Transformers perform well
    - The only model with embeddings for **numerical features**
- GBDTs process **numerical features** via thresholds

*(2022) On Embeddings for Numerical Features in Tabular Deep Learning*

# Moreover…

- Transformers perform well
  - The only model with embeddings for **numerical features**
- GBDTs process **numerical features** via thresholds
- MLP is a universal approximator in theory…

*(2022) On Embeddings for Numerical Features in Tabular Deep Learning*

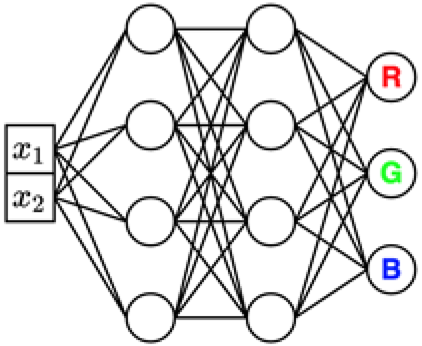
# Moreover…

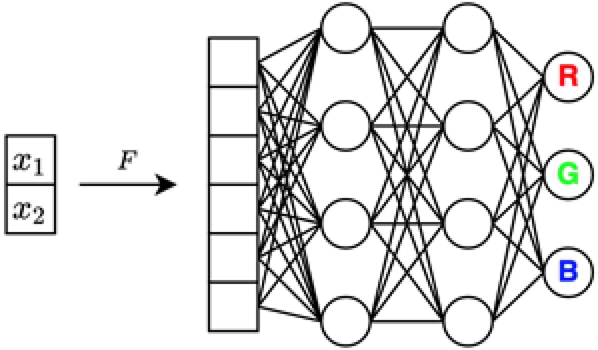
- Transformers perform well
  - The only model with embeddings for **numerical features**
- GBDTs process **numerical features** via thresholds
- MLP is a universal approximator in theory…
- … but not in practice. Though, **changing the input space can help**
  - "Fourier Features Let Networks Learn High Frequency Functions in Low Dimensional Domains" (Matthew Tancik et al., 2020)
  - "NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis" (Ben Mildenhall et al., 2020)

*(2022) On Embeddings for Numerical Features in Tabular Deep Learning*

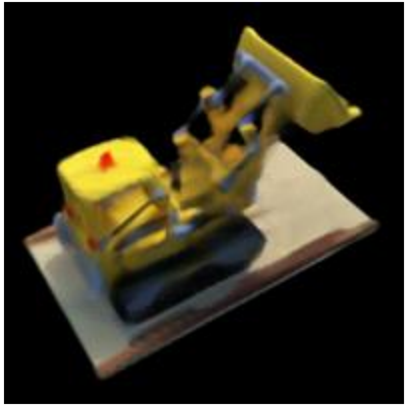
# Input representation matters

*Fourier Features Let Networks Learn High Frequency Functions in Low Dimensional Domains (Tancik et al., NeurIPS 2020)*




The original image



# Moreover…

- Transformers perform well
  - The only model with embeddings for **numerical features**
- GBDTs process **numerical features** via thresholds
- MLP is a universal approximator in theory…
- … but not in practice. Though, **changing the input space can help**
  - “Fourier Features Let Networks Learn High Frequency Functions in Low Dimensional Domains” (Matthew Tancik et al., 2020)
  - “NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis” (Ben Mildenhall et al., 2020)
- Little work on **numerical features** processing

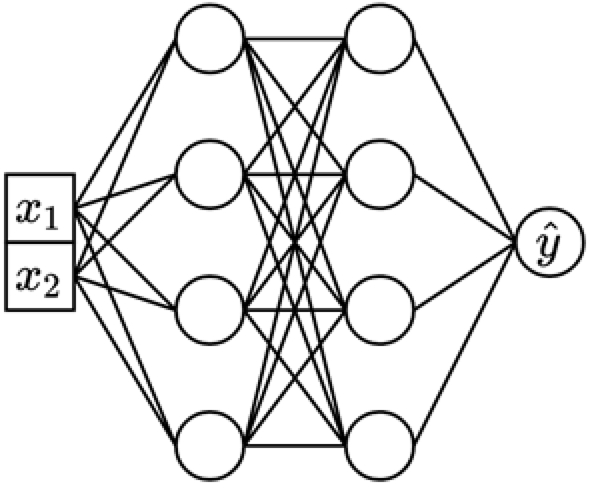
*(2022) On Embeddings for Numerical Features in Tabular Deep Learning*

# Questions

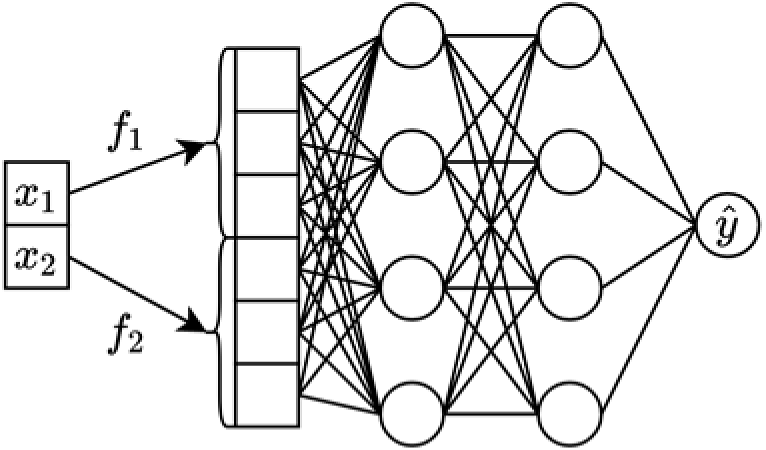
- Can we improve the way numerical features are processed?
- Can MLP-like models benefit from embeddings for numerical features?

*(2022) On Embeddings for Numerical Features in Tabular Deep Learning*

# MLP with embeddings



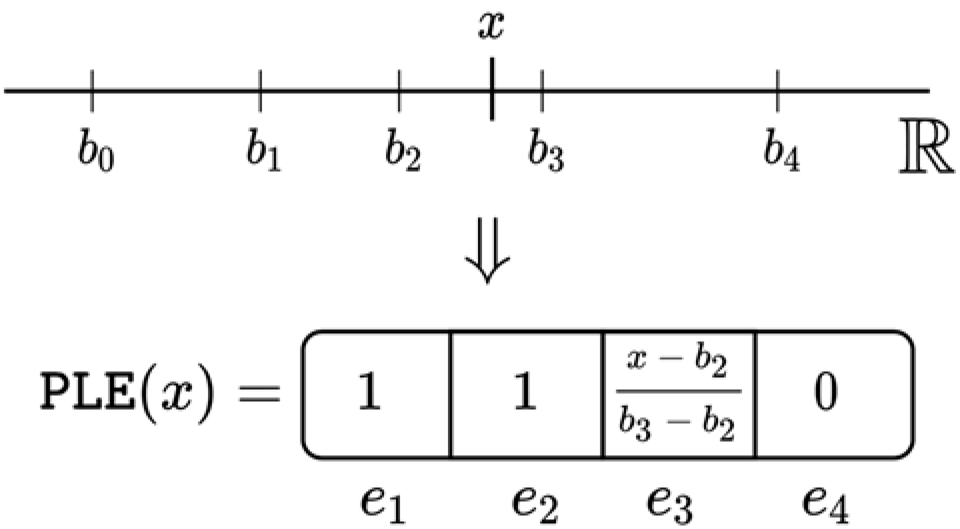

Without embeddings




With embeddings

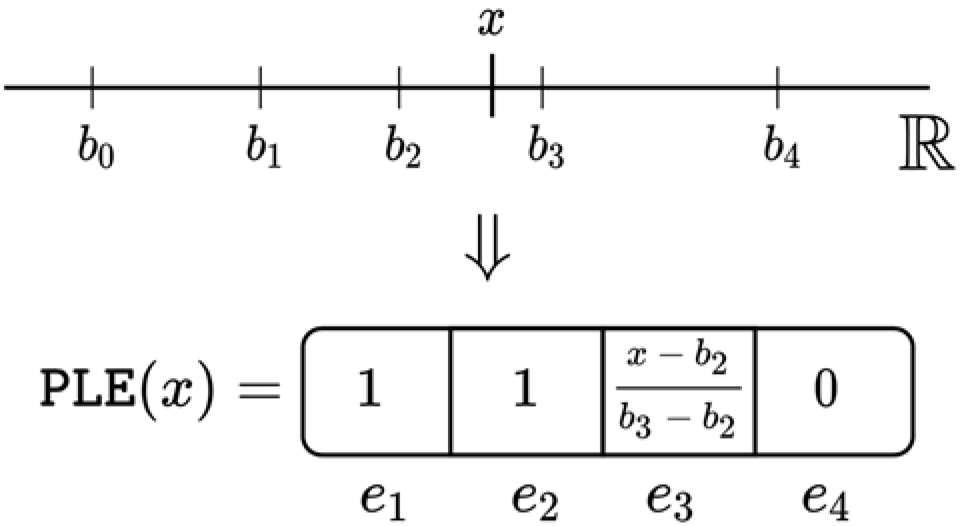
*(2022) On Embeddings for Numerical Features in Tabular Deep Learning*

# Piecewise-linear encoding




*(2022) On Embeddings for Numerical Features in Tabular Deep Learning*

# Piecewise-linear encoding




For Transformer-based models:

- $v_t$ - the embedding of the t-th bin

$$f_i(x) = v_0 + \sum_{t=1}^{T} e_t \cdot v_t = \mathtt{Linear}(\mathtt{PLE}(x))$$

*(2022) On Embeddings for Numerical Features in Tabular Deep Learning*

# Piecewise-linear encoding

Quantile binning

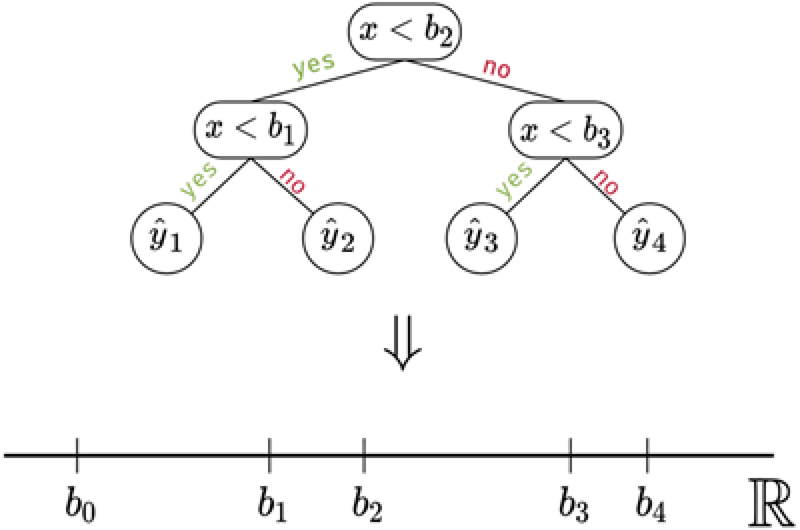
$$b_t = \mathrm{Q}_{\frac{t}{T}}\left(\{x_i^{j(num)}\}_{j \in J_{train}}\right)$$

# Piecewise-linear encoding

Quantile binning

$$b_t = \mathrm{Q}_{\frac{t}{T}}\left(\{x_i^{j(num)}\}_{j \in J_{train}}\right)$$

Target-aware binning




*(2022) On Embeddings for Numerical Features in Tabular Deep Learning*

# Periodic activation functions

- (this approach is unrelated to PLE)
- Inspired by the success of periodic functions in other fields

$$f_i(x) = \texttt{Periodic}(x) = \texttt{concat}[\sin(v), \cos(v)]$$
$$v = [2\pi c_1 x, \ldots, 2\pi c_k x]$$

*(2022) On Embeddings for Numerical Features in Tabular Deep Learning*

# Other approaches

- Stacking “conventional” layers (linear, ReLU, SoftMax, …)
- Stacking “conventional” layers on top of PLE or Periodic

*(2022) On Embeddings for Numerical Features in Tabular Deep Learning*

# Model names

| Embedding name | Embedding function f_i | Comment |
| --- | --- | --- |
| L | Linear(x) | |
| LR | ReLU(Linear(x)) | |
| Q-LR | ReLU(Linear(PLE(x))) | quantile-based PLE |
| T-LR | ReLU(Linear(PLE(x))) | target-based PLE |
| PLR | ReLU(Linear(Periodic(x))) | The “LR” addition is more important, than for PLE |

Model name = <Backbone-Embedding>
Examples:

- Transformer-L (== FT-Transformer)
- MLP-PLR

*(2022) On Embeddings for Numerical Features in Tabular Deep Learning*

# Experiments: datasets and protocol

| Dataset | N | K | Metric |
|---|---|---|---|
| Gesture | 10K | 32 | Accuracy (M) |
| Churn modelling | 10K | 11 | Accuracy (B) |
| Eye movements | 11K | 26 | Accuracy (M) |
| California Housing | 21K | 8 | RMSE |
| House pricing | 23K | 16 | RMSE |
| Adult income | 49K | 14 | Accuracy (B) |
| Otto products | 62K | 93 | Accuracy (M) |
| Higgs (small) | 98K | 28 | Accuracy (B) |
| FB comments | 197K | 51 | RMSE |
| Santander | 200K | 200 | Accuracy (M) |
| Covertype | 581K | 54 | Accuracy (M) |
| Microsoft | 1201K | 136 | RMSE |

*N ~ dataset size*
*K ~ number of features*

*B ~ binary*
*M ~ multiclass*

- Tuning
  - mostly Optuna (Akiba et al., 2019) (50-100 iterations)
- Evaluation
  - 15 random seeds
  - ensembles: three ensembles (each consists of five single models)
- No DL tricks
  - no augmentation
  - no lr scheduling
  - no pretraining
  - etc.

*(2022) On Embeddings for Numerical Features in Tabular Deep Learning*

# Experiments: results

| Model | Average rank (std.) |
|---|---|
| CatBoost | 6.8 (4.9) |
| XGBoost | 9.0 (5.7) |
| MLP | 15.6 (2.4) |
| MLP-LR | 10.2 (4.4) |
| MLP-Q-LR | 10.7 (4.6) |
| MLP-T-LR | 10.3 (3.8) |
| MLP-PLR | 4.9 (4.8) |
| Transformer-L | 10.6 (3.3) |
| Transformer-LR | 9.4 (4.1) |
| Transformer-Q-LR | 8.5 (5.5) |
| Transformer-T-LR | 7.2 (4.6) |
| Transformer-PLR | 6.0 (4.5) |

- The benchmark is biased towards GBDT-friendly problems
- MLP-LR is consistently better than MLP

*(2022) On Embeddings for Numerical Features in Tabular Deep Learning*

# Experiments: results

| Model | Average rank (std.) |
|---|---|
| CatBoost | 6.8 (4.9) |
| XGBoost | 9.0 (5.7) |
| MLP | 15.6 (2.4) |
| MLP-LR | 10.2 (4.4) |
| MLP-Q-LR | 10.7 (4.6) |
| MLP-T-LR | 10.3 (3.8) |
| MLP-PLR | 4.9 (4.8) |
| Transformer-L | 10.6 (3.3) |
| Transformer-LR | 9.4 (4.1) |
| Transformer-Q-LR | 8.5 (5.5) |
| Transformer-T-LR | 7.2 (4.6) |
| Transformer-PLR | 6.0 (4.5) |

- The benchmark is biased towards GBDT-friendly problems
- MLP-LR is consistently better than MLP

**Embeddings for numerical features:**

- **can provide significant boost**
- **are applicable to MLP-like models**
  - See MLP vs MLP-PLR!
- **allow MLP-like models to compete with Transformer**

*(2022) On Embeddings for Numerical Features in Tabular Deep Learning*

# Conclusion

- Backbones
  - MLP is a great backbone for researchers and practitioners
  - ResNet may (or may not) provide an extra bit of performance
  - Transformers are competitive, but slow (unclear if it is worth it)
- Embeddings for numerical features
  - can provide significant performance boost
  - `Linear + ReLU`
    - low risk & low reward
  - `Periodic + Linear + ReLU`
    - tune sigma: [0.01, 0.02, 0.05, 0.1, 0.5, 1.0, …]
    - for other hyperparameters, take inspiration from the official repository
  - PLE-based solutions can also provide good performance

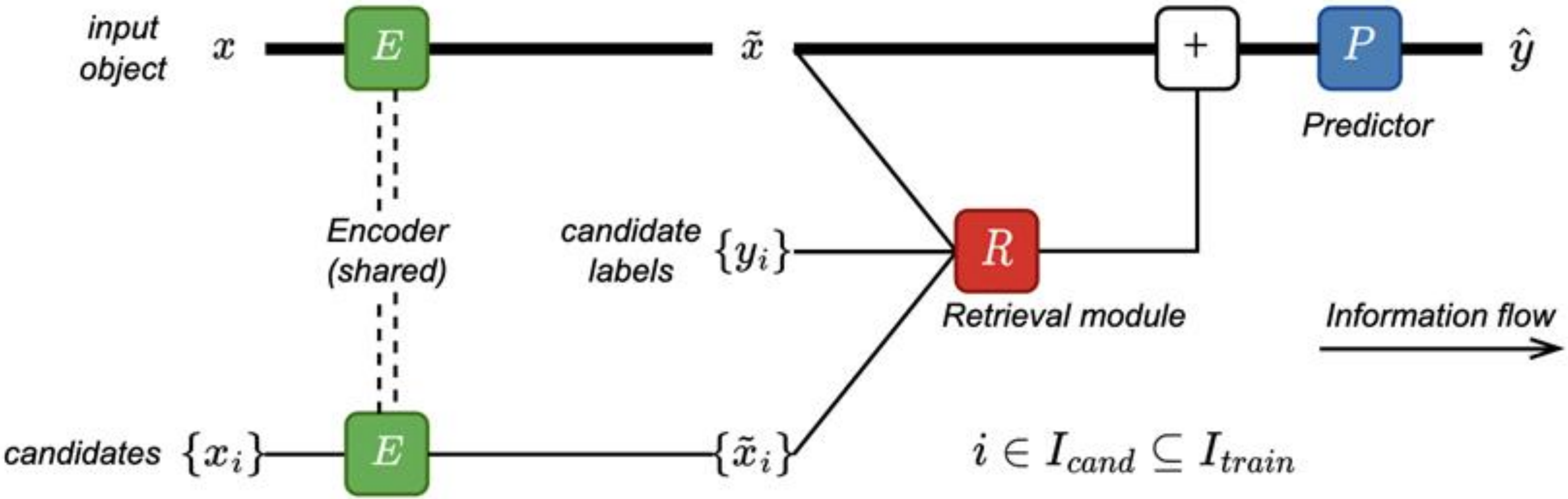
# Outline

- Introduction
- The pre-deep learning era of Tabular ML
- Modern Tabular Deep Learning
    - TabR
- Real-world impact

# Retrieval-Augmented Learning

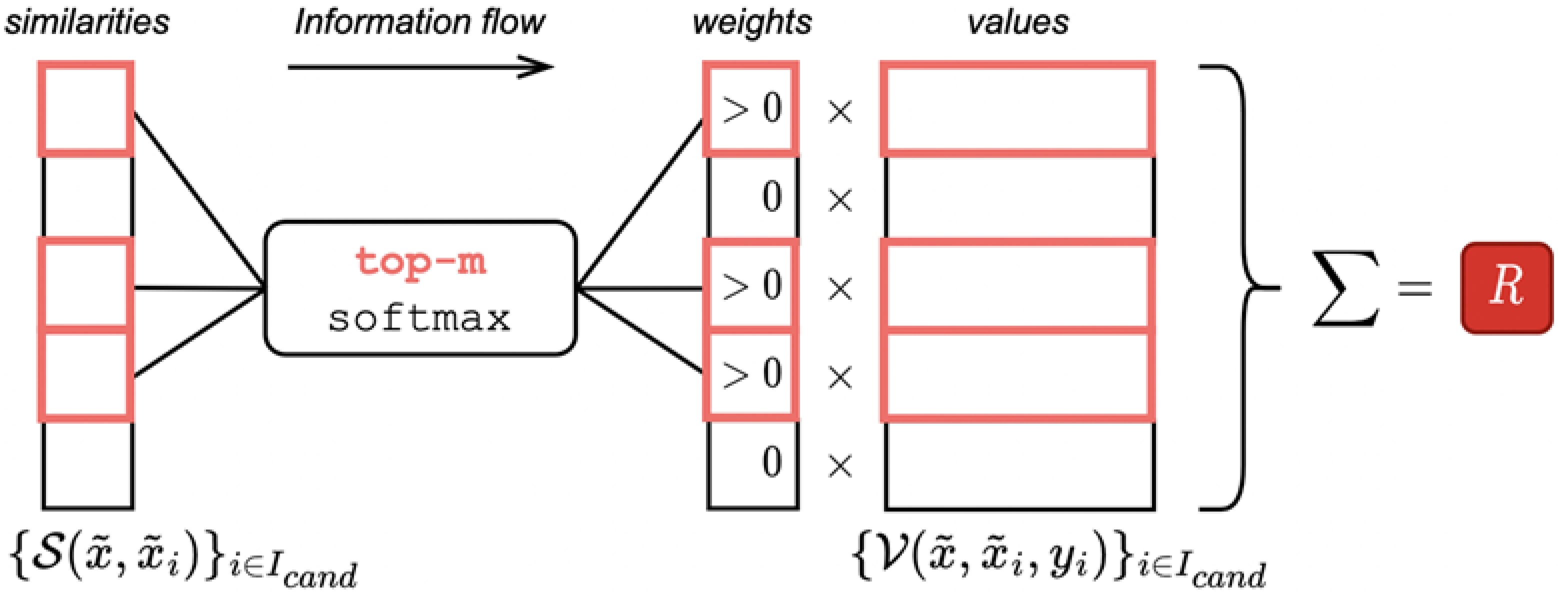
- Is originally motivated by the local learning paradigm (Vapnik et al. 1992)
- Demonstrates success in NLP and computer vision tasks
- Provides higher interpretability and robustness

# TabR




*TabR: Tabular Deep Learning Meets Nearest Neighbors (ICLR 2024)*

# TabR




*TabR: Tabular Deep Learning Meets Nearest Neighbors (ICLR 2024)*

# Technical insights

The retrieval module R

- Linear complexity w.r.t. the number of candidates
- The inter-object communication happens only once

The similarity module S

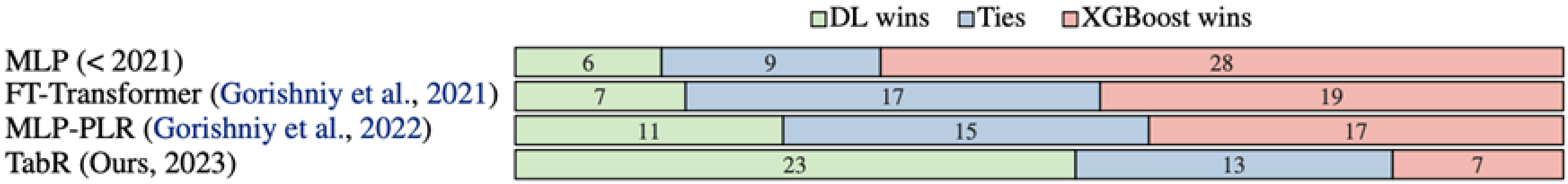
- By default, the L2 distance is recommended (important!)

The value module V

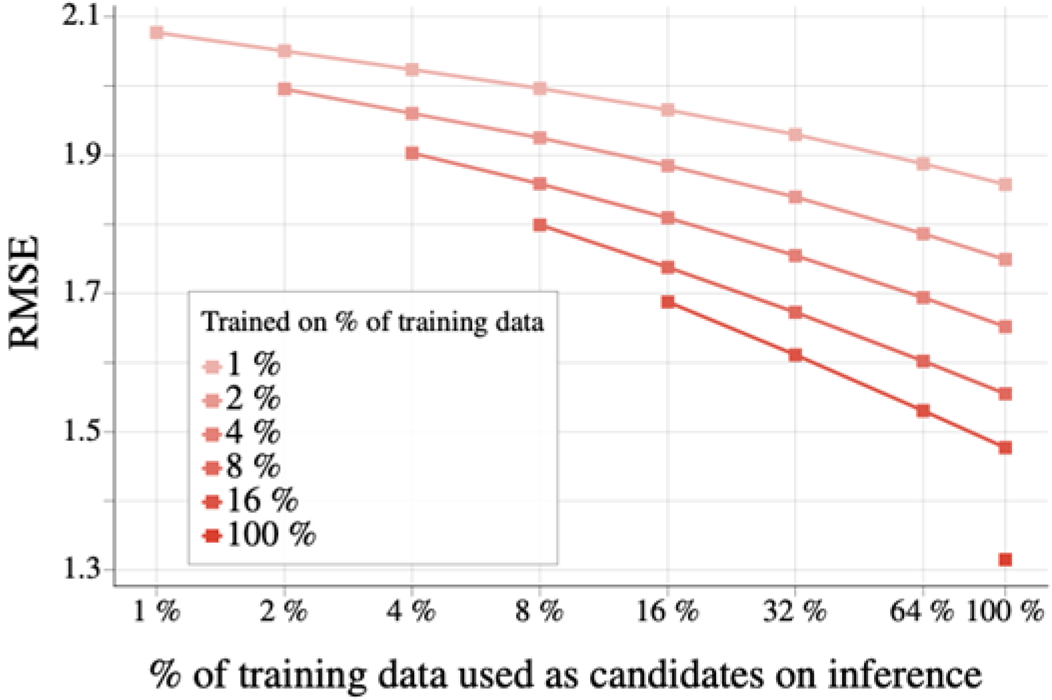
- Can depend on objects and their interactions

*TabR: Tabular Deep Learning Meets Nearest Neighbors (ICLR 2024)*

# TabR results



# Training on a subset of data



# Limitations

- Reminder: simple ML models suffer from distributions shifts in features and/or labels of individual objects.
- Retrieval-based models also suffer from distribution shifts in interactions between objects.
- To prevent such problems, one has to think how to configure the retrieval behavior in each individual use case.

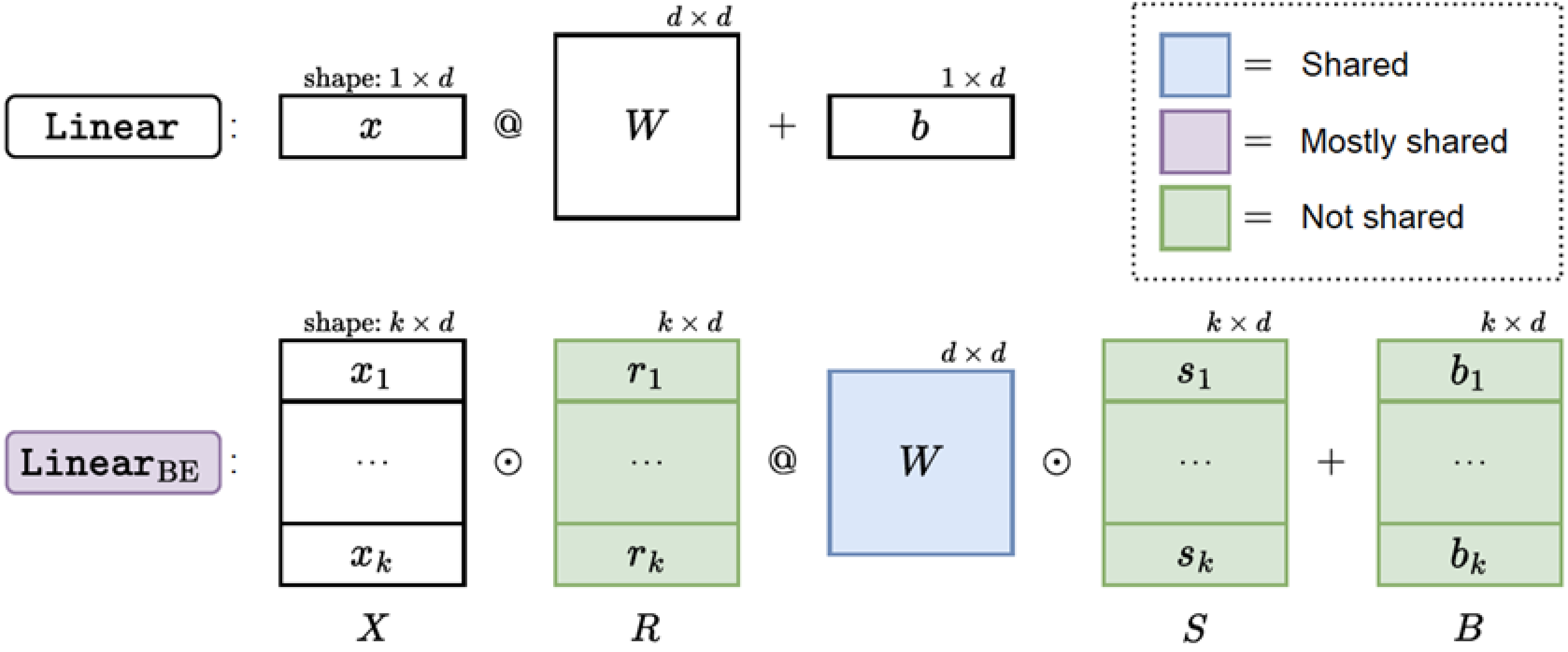
# Outline

- Introduction
- The pre-deep learning era of Tabular ML
- Modern Tabular Deep Learning
    - TabM
- Real-world impact

# Ensembles of Models in Machine Learning

- Main idea: train several models and combine predictions from them
- GBDT are essentially an ensemble
- Go-to recipe in DL: train several *independent* models and average the predictions
    - Can be used for any model
    - Often improves accuracy
    - Higher memory and runtime costs

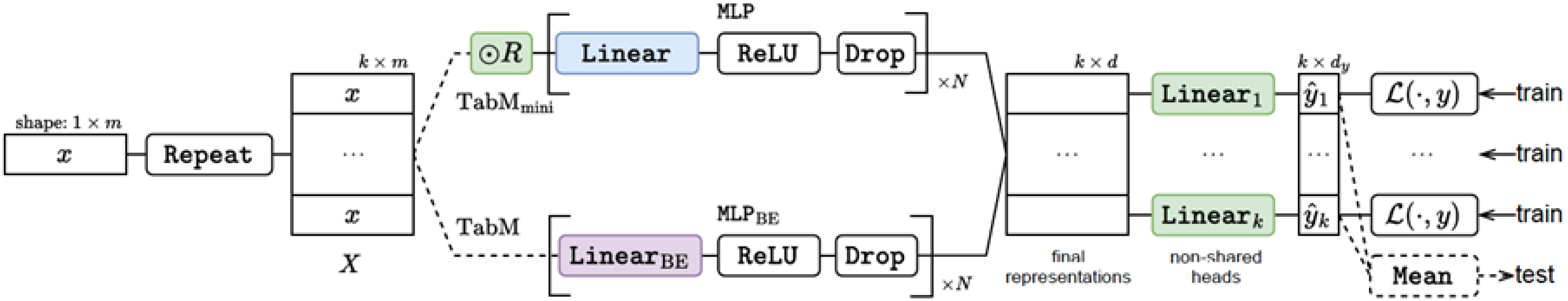
# BatchEnsemble (Wen et al., 2020): main idea




R, S, B - *adapters*

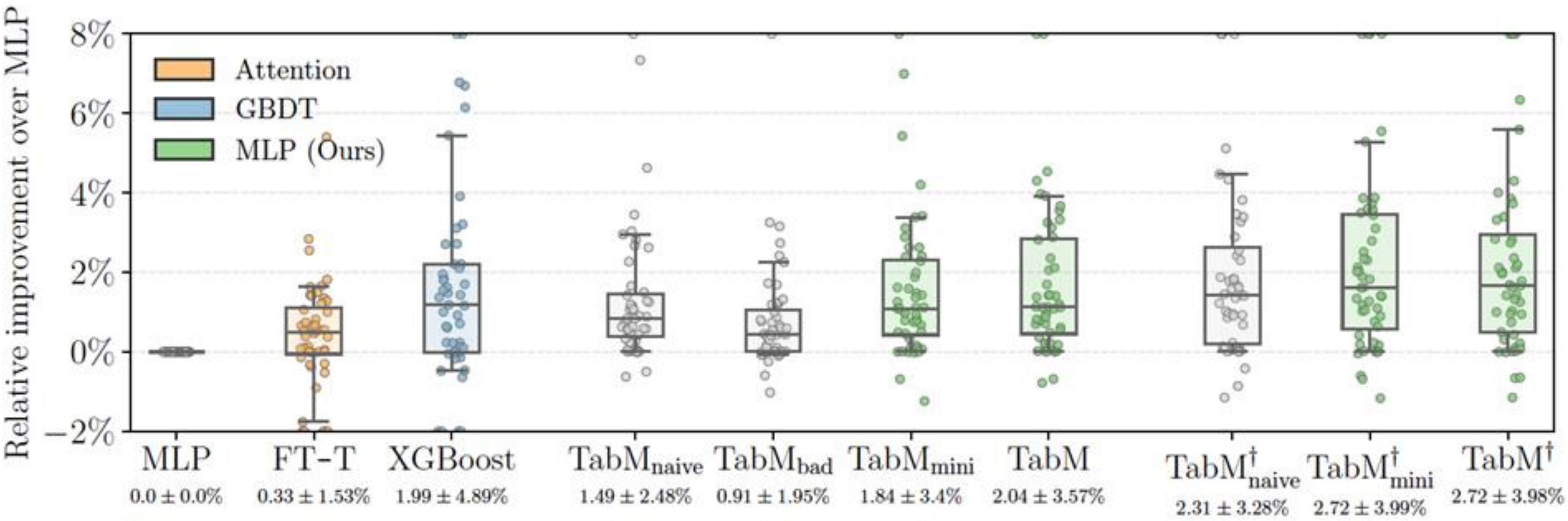
Since k << d, runtime and memory overhead are tolerable!

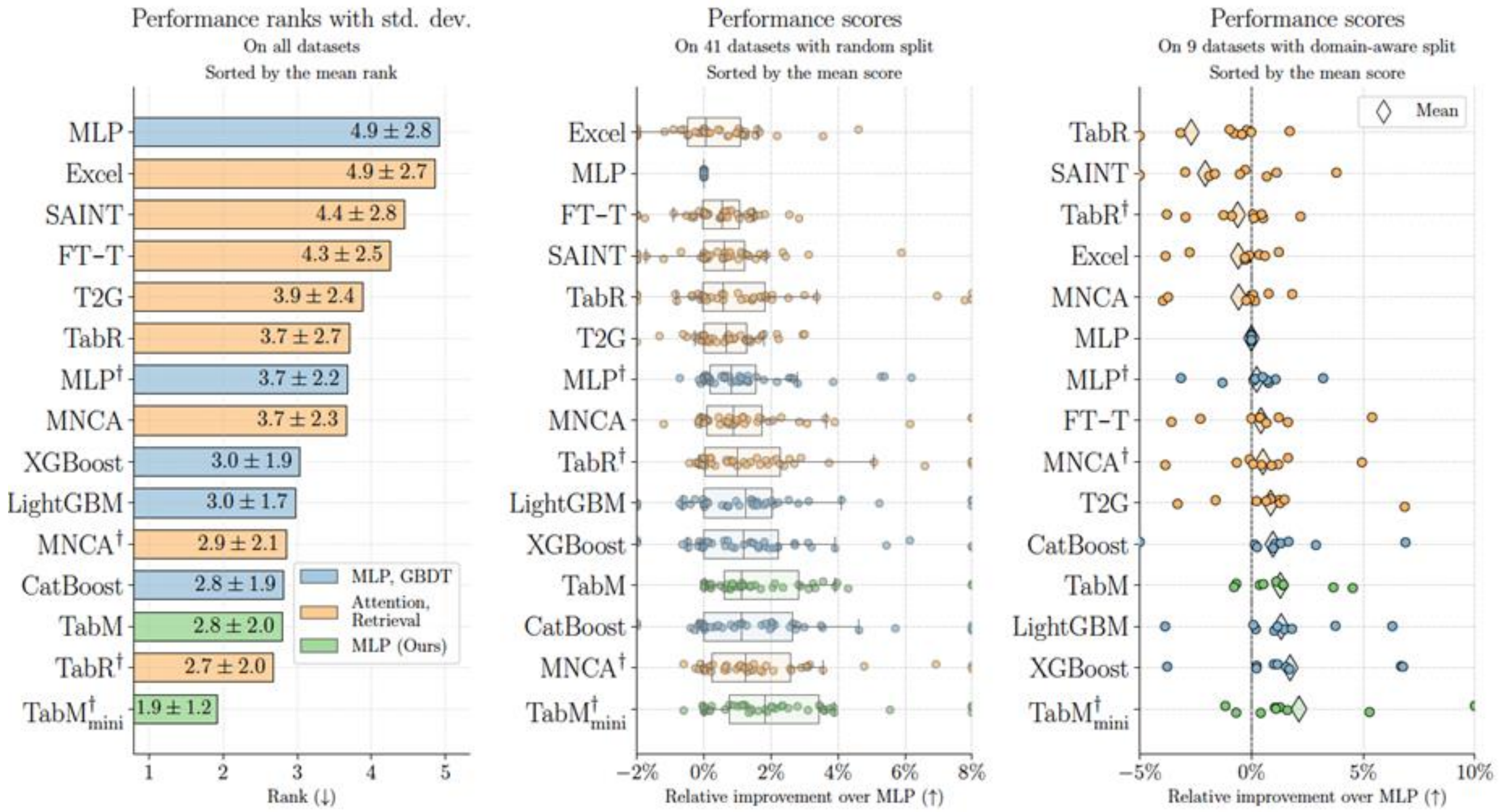
# TabM: BatchEnsemble meets Tabular DL




- TabM with k = 1 is equivalent to MLP
- Specific initialization of adapters is needed
- Can be combined with non-linear feature embeddings

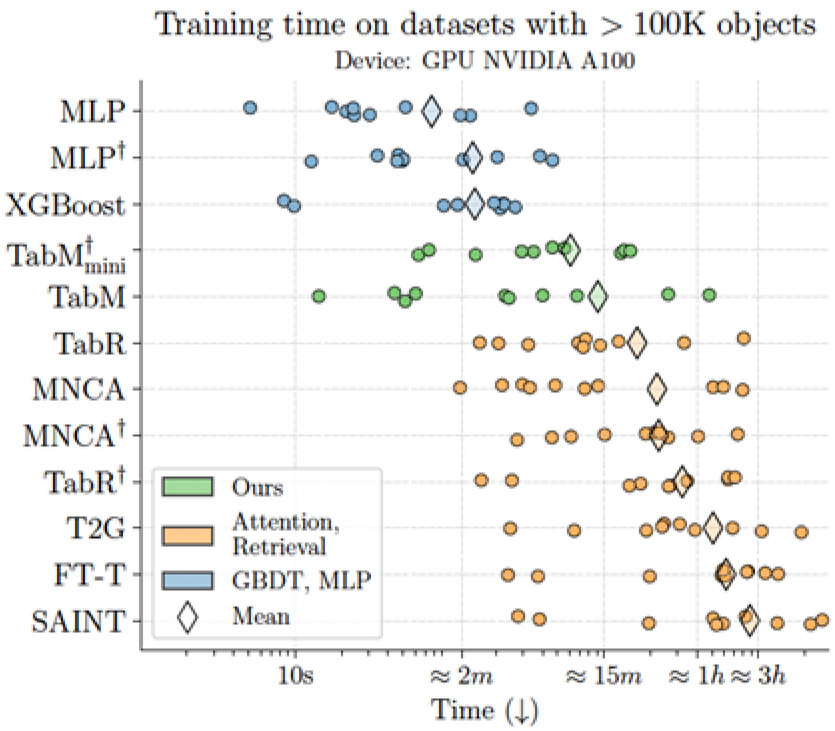
# TabM: results



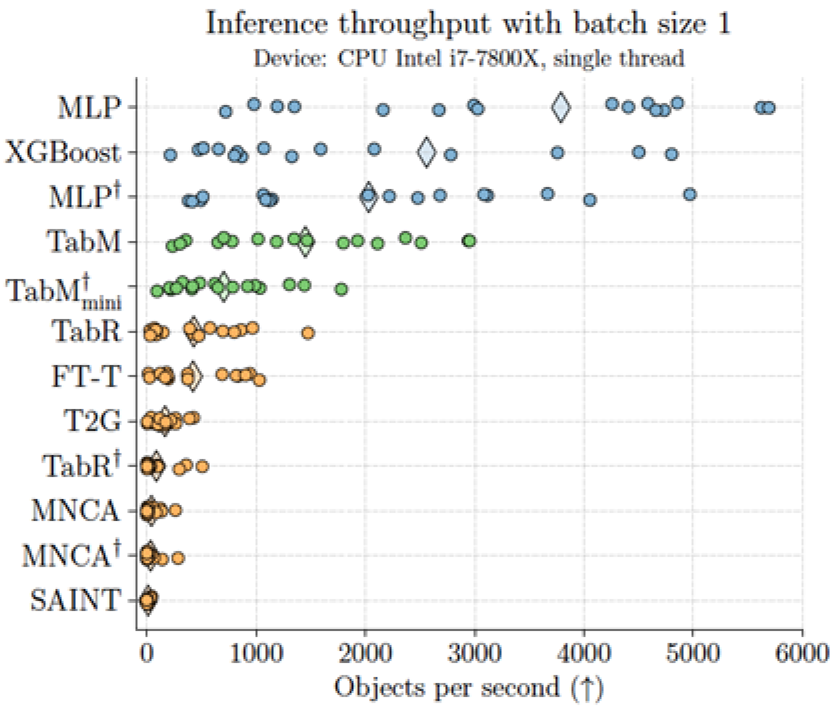


Performance ranks with std. dev.
On all datasets
Sorted by the mean rank
MLP 4.9 ± 2.8
Excel 4.9 ± 2.7
SAINT 4.4 ± 2.8
FT-T 4.3 ± 2.5
T2G 3.9 ± 2.4
TabR 3.7 ± 2.7
MLP† 3.7 ± 2.2
MNCA 3.7 ± 2.3
XGBoost 3.0 ± 1.9
LightGBM 3.0 ± 1.7
MNCA† 2.9 ± 2.1
CatBoost 2.8 ± 1.9
TabM 2.8 ± 2.0
TabR† 2.7 ± 2.0
TabM†mini 1.9 ± 1.2
MLP, GBDT
Attention, Retrieval
MLP (Ours)
Rank (↓)
Performance scores
On 41 datasets with random split
Sorted by the mean score
Excel
MLP
FT-T
SAINT
TabR
T2G
MLP†
MNCA
TabR†
LightGBM
XGBoost
TabM
CatBoost
MNCA†
TabM†mini
Relative improvement over MLP (↑)
Performance scores
On 9 datasets with domain-aware split
Sorted by the mean score
Mean
TabR
SAINT
TabR†
Excel
MNCA
MLP
MLP†
FT-T
MNCA†
T2G
CatBoost
TabM
LightGBM
XGBoost
TabM†mini
Relative improvement over MLP (↑)

# Efficiency

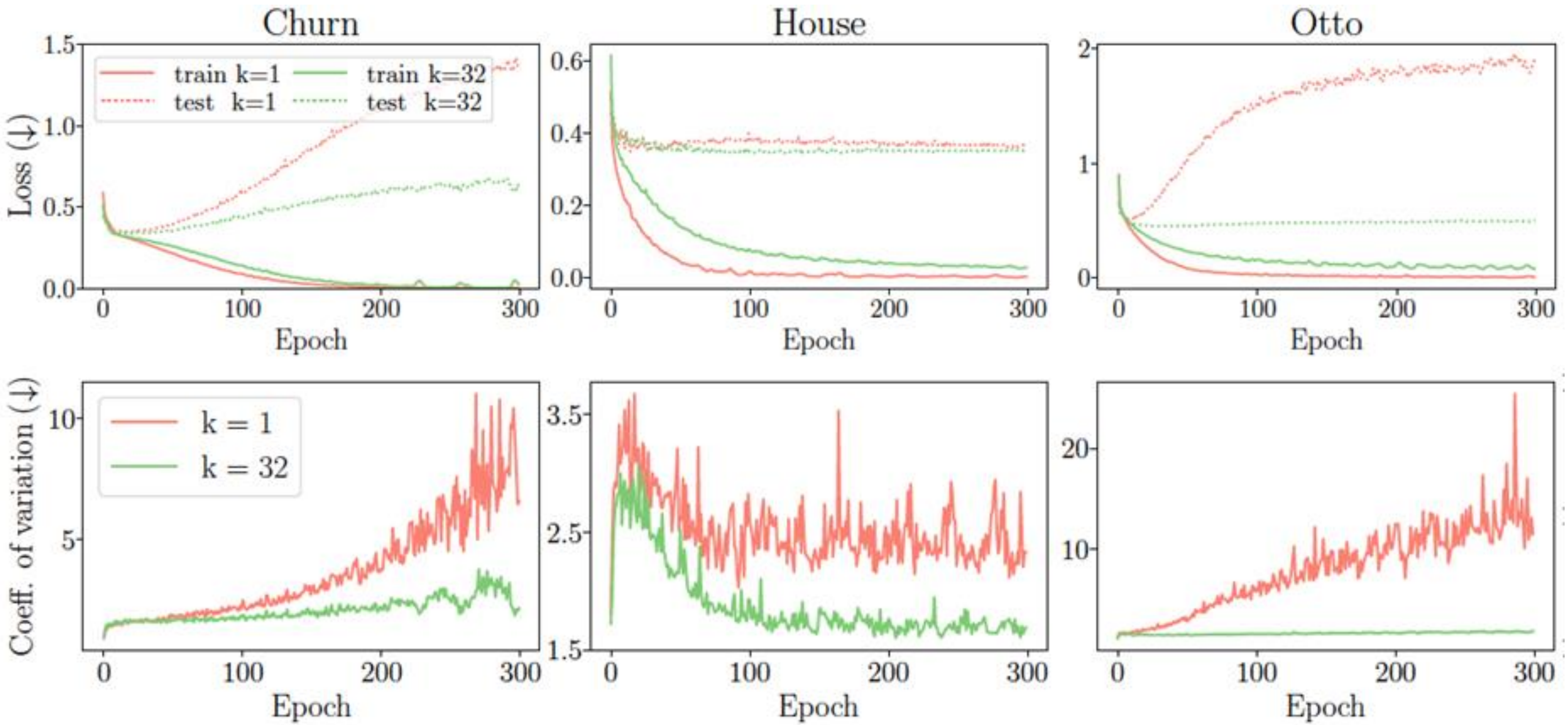


Training time on datasets with > 100K objects
Device: GPU NVIDIA A100
MLP
MLP†
XGBoost
TabM†mini
TabM
TabR
MNCA
MNCA†
TabR†
T2G
FT-T
SAINT
Ours
Attention, Retrieval
GBDT, MLP
Mean
10s
≈ 2m
≈ 15m
≈ 1h
≈ 3h
Time (↓)

Inference throughput with batch size 1
Device: CPU Intel i7-7800X, single thread
MLP
XGBoost
MLP†
TabM
TabM†mini
TabR
FT-T
T2G
TabR†
MNCA
MNCA†
SAINT
0
1000
2000
3000
4000
5000
6000
Objects per second (↑)

# Optimization properties of TabM



# Conclusion

- TabM with non-linear feature embeddings are currently the state-of-the-art
- TabM typically outperforms GBDT on existing benchmarks
- TabM exhibits stable optimization and less overfitting

# Outline

- Introduction
- The pre-deep learning era of Tabular ML
- Modern Tabular Deep Learning
- Real-world impact

# Tabular DL in our lives

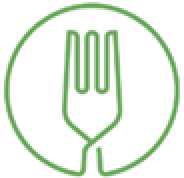
**DeepETA: How Uber Predicts Arrival Times Using Deep Learning**

February 10, 2022 / Global

Rides

Freight

Eats



**How we built it: Stripe Radar**

Our most recent architecture evolution occurred in mid-2022 when we migrated from an ensemble "Wide & Deep model," composed of an XGBoost model and a deep neural network (DNN), to a pure DNN-only model. The result was a model that trains faster, scales better, and is more adaptable to the most cutting-edge ML techniques.

**В ННГУ усовершенствовали нейросеть для диагностики скорости старения**

Ученые Университета Лобачевского усовершенствовали нейросеть для диагностики скорости старения. Новая модель иммунологических часов получила название SImAge (Small Immuno Age). Она построена на основе глубокой нейронной сети FT-Transformer. Нейросеть оценивает состояние организма по 10 биомаркерам, которые отражают ...

**Challenging Gradient Boosted Decision Trees with Tabular Transformers for Fraud Detection at Booking.com**

# Conclusion

- Tabular DL is extremely impactful research field with many unresolved questions
- New models are being developed and the progress has not converged
- GBDTs are still in wide use but their primacy has been challenged
- *Tomorrow:* Advanced topics in Tabular DL

# Questions?

# Advanced Topics in Tabular Deep Learning

Lecturer: Artem Babenko

Research

ASCOMP 2024

# Outline

- Quick recap
- Tabular Benchmarks
- Pretraining in Tabular DL
- Cross-domain learning
- Generative tabular models
- Future directions

# Outline

- Quick recap
- Tabular Benchmarks
- Pretraining in Tabular DL
- Cross-domain learning
- Generative tabular models
- Future directions

# Recap from yesterday

- Tabular problems are everywhere
- “Shallow” GBDT models are still a popular choice
- Tabular DL architectures are actively developed
- Are new DL architectures the only research direction?
  - No!

# Outline

- Quick recap
- Tabular Benchmarks
- Pretraining in Tabular DL
- Cross-domain learning
- Generative tabular models
- Future directions

# Where do tabular DL researchers get datasets?

- openml.org
- archive.ics.uci.edu
- kaggle.com/datasets
- `from sklearn.datasets import *`
- Do we care to examine those 10-20-100 datasets? - Rarely!

# Let’s Look at the Academic Benchmarks

| Dataset | #Samples | #Features | Citation | Not a Real World Task | Time split impossible, or not used (if needed) | Leak | Not Tabular | Synthetic Unkown origin | GOV Records | Ques |
|---|---|---|---|---|---|---|---|---|---|---|
| lymph | 148 | 19 | https://www.openml.org/search?type=data&status=active&id=10 | 1 | 0 | 0 | 0 | 0 | 0 | |
| qsar-biodeg | 155 | 42 | https://www.openml.org/search?type=data&status=active&id=1494 | 0 | 0 | 0 | 1 | 0 | 0 | |
| audiology | 226 | 70 | https://archive.ics.uci.edu/dataset/8/audiology+standardized | 1 | 0 | 0 | 0 | 0 | 0 | |
| heart-h | 294 | 14 | https://openml.org/search?type=data&status=active&id=51 | 0 | 0 | 0 | 0 | 0 | 0 | |
| colic | 368 | 27 | https://www.openml.org/search?type=data&status=active&id=25 | 1 | 0 | 0 | 0 | 1 | 0 | |
| monks-problems-2 | 601 | 7 | https://www.openml.org/search?type=data&status=active&id=334 | 1 | 0 | 0 | 0 | 1 | 0 | |
| balance-scale | 625 | 5 | http://archive.ics.uci.edu/dataset/12/balance+scale | 1 | 0 | 0 | 0 | 1 | 0 | |
| profb | 672 | 10 | https://www.openml.org/search?type=data&status=active&id=470 | 1 | 0 | 0 | 0 | 0 | 0 | |
| Australian | 690 | 15 | https://archive.ics.uci.edu/dataset/143/statlog+australian+credit+approval | 0 | 1 | 0 | 0 | 0 | 0 | |
| credit-approval | 690 | 16 | https://archive.ics.uci.edu/dataset/27/credit+approval | 0 | 1 | 0 | 0 | 1 | 0 | |
| vehicle | 846 | 19 | https://www.openml.org/search?type=data&status=active&id=54 | 0 | 0 | 0 | 1 | 1 | 0 | |
| cnae-9 | 1080 | 857 | https://www.openml.org/search?type=data&status=active&id=1468 | 0 | 0 | 0 | 1 | 0 | 0 | |
| socmob | 1156 | 6 | https://www.openml.org/search?type=data&status=active&id=44987 | 0 | 1 | 0 | 0 | 1 | 1 | |
| 100-plants-texture | 1599 | 65 | https://archive.ics.uci.edu/dataset/241/one+hundred+plant+species+leaves+data+set | 1 | 0 | 0 | 1 | 0 | 0 | |

benchmark critique why ours tabpfn

# What did we find?

## Problems:

**Data Leakage (10 datasets).** data-leaks stemming from data preparation errors, or inappropriate data splits being used in papers using the datasets.

**No time data available (most datasets).** either represent a fixed snapshot of some real-world phenomena, or don't have a way to construct a time-based validation/test sets

**Dataset Duplication** (California Housing, House 16H, house_sales, kdd_ipums_la_97-small, houses) - all datasets are from 1990 census data

**Dataset Size.** 19/100 less than 10k samples.

**Synthetic data.** (or from an unknown source). datasets for which the original data source is untraceable.

**Not Tabular.** datasets where underlying data is not tabular like images, audio, text or graphs

**>50%**

Datasets don't handle time properly

**38%**

"Problematic" Datasets

**~20**

Features available

**<1kk**

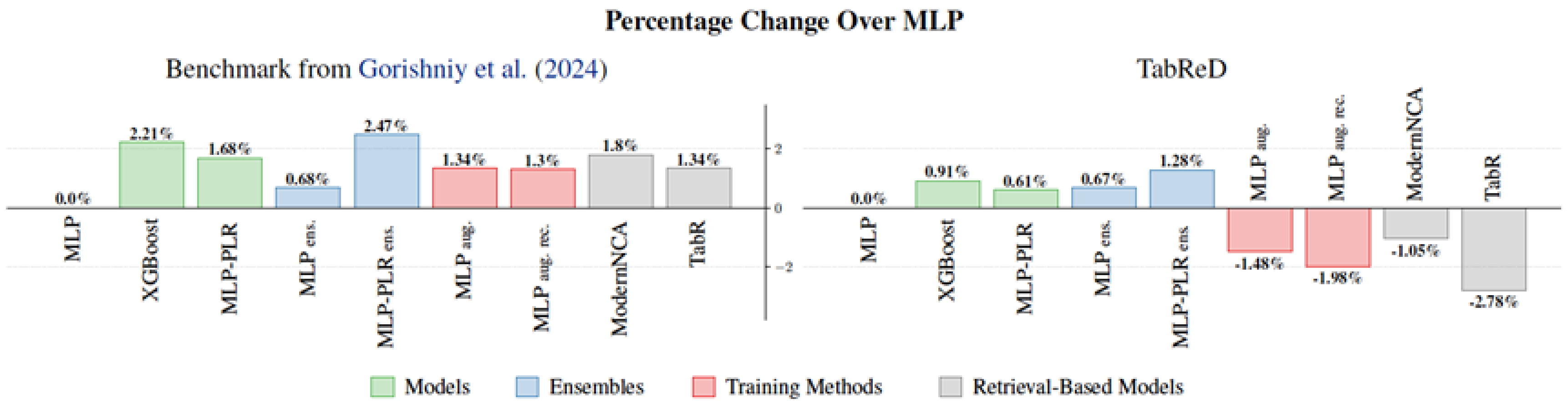
Small sample sizes
Majority is bellow 100k samples

# TabRed: focus on temporal-shift based evaluation

| Benchmark | Dataset Sizes ($Q_{50}$) | | Issues (#Issues / #Datasets) | | | Time-split | | |
|---|---|---|---|---|---|---|---|---|
| | #Samples | #Features | Data-Leakage | Synthetic or Untraceable | Non-Tabular | Needed | Possible | Used |
| Grinsztajn et al. [22] | 16,679 | 13 | 7 / 44 | 1 / 44 | 7 / 44 | 22 | 5 | |
| Tabzilla [40] | 3,087 | 23 | 3 / 36 | 6 / 36 | 12 / 36 | 12 | 0 | |
| WildTab [35] | 546,543 | 10 | 1* / 3 | 1 / 3 | 0 / 3 | 1 | 1 | ✗ |
| TableShift [18] | 840,582 | 23 | 0 / 15 | 0 / 15 | 0 / 15 | 15 | 8 | |
| Gorishniy et al. [21] | 57,909 | 20 | 1* / 10 | 1 / 10 | 0 / 10 | 7 | 1 | |
| **TabReD** (ours) | 7,163,150 | 261 | ✗ | ✗ | ✗ | ✓ | ✓ | ✓ |

| Methods | Classification (ROC AUC ↑) | | | Regression (RMSE ↓) | | | | | Average Rank |
|---|---|---|---|---|---|---|---|---|---|
| | Homesite Insurance | Ecom Offers | HomeCredit Default | Sberbank Housing | Cooking Time | Delivery ETA | Maps Routing | Weather | |
| **Classical ML Baselines** | | | | | | | | | |
| XGBoost | 0.9601 | 0.5763 | **0.8670** | <u>0.2419</u> | 0.4823 | <u>0.5468</u> | <u>0.1616</u> | <u>1.4671</u> | **2.6 ± 1.2** |
| LightGBM | 0.9603 | 0.5758 | <u>0.8664</u> | 0.2468 | 0.4826 | <u>0.5468</u> | 0.1618 | **1.4625** | **2.9 ± 1.2** |
| CatBoost | 0.9606 | 0.5596 | 0.8621 | 0.2482 | 0.4823 | **0.5465** | 0.1619 | <u>1.4688</u> | **3.1 ± 1.4** |
| RandomForest | 0.9570 | 0.5764 | 0.8269 | 0.2640 | 0.4884 | 0.5959 | 0.1653 | 1.5838 | 7.1 ± 2.0 |
| Linear | 0.9290 | 0.5665 | 0.8168 | 0.2509 | 0.4882 | 0.5579 | 0.1709 | 1.7679 | 8.1 ± 2.5 |
| **Tabular DL Models** | | | | | | | | | |
| MLP | 0.9500 | <u>0.6015</u> | 0.8545 | 0.2508 | 0.4820 | 0.5504 | 0.1622 | 1.5470 | 4.8 ± 1.7 |
| SNN | 0.9492 | 0.5996 | 0.8551 | 0.2858 | 0.4838 | 0.5544 | 0.1651 | 1.5649 | 6.4 ± 1.9 |
| DCNv2 | 0.9392 | 0.5955 | 0.8466 | 0.2770 | 0.4842 | 0.5532 | 0.1672 | 1.5782 | 7.4 ± 2.3 |
| ResNet | 0.9469 | 0.5998 | 0.8493 | 0.2743 | 0.4825 | 0.5527 | 0.1625 | 1.5021 | 5.5 ± 2.1 |
| FT-Transformer | <u>0.9622</u> | 0.5775 | 0.8571 | 0.2440 | 0.4820 | 0.5542 | 0.1625 | 1.5104 | 4.4 ± 1.4 |
| MLP-PLR | <u>0.9621</u> | 0.5957 | 0.8568 | 0.2438 | <u>0.4812</u> | 0.5527 | <u>0.1616</u> | 1.5177 | **3.6 ± 1.5** |
| Trompt | 0.9546 | 0.5792 | 0.8381 | 0.2596 | 0.4834 | 0.5563 | 0.1652 | 1.5722 | 6.8 ± 2.0 |
| **Retrieval Augmented Tabular DL** | | | | | | | | | |
| TabR-S | 0.9487 | 0.5943 | 0.8501 | 0.2820 | 0.4828 | 0.5514 | 0.1639 | <u>1.4666</u> | 5.8 ± 2.2 |
| ModernNCA | 0.9514 | 0.5765 | 0.8531 | 0.2593 | 0.4825 | 0.5498 | 0.1625 | 1.5062 | 5.0 ± 1.3 |

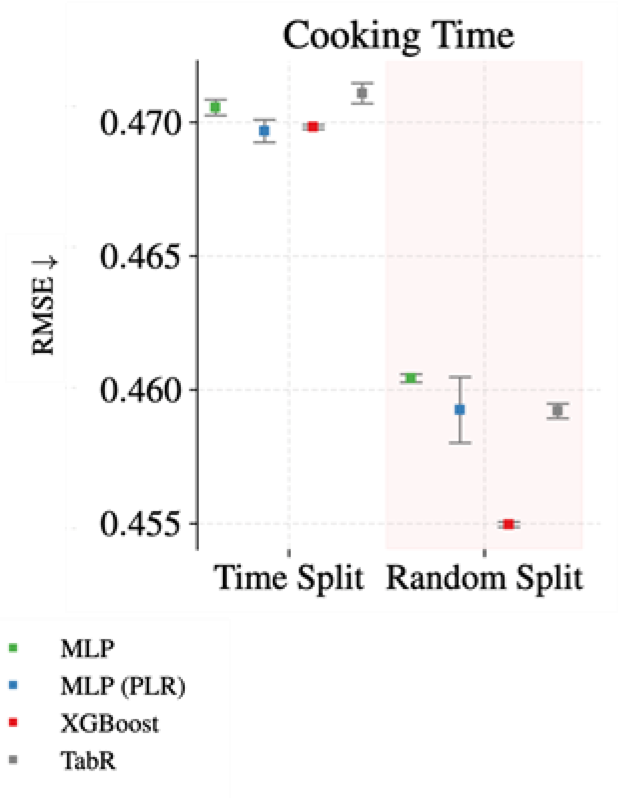
# Findings on TabReD




- Performance differences are less pronounced (feature engineering)
- Non-linear feature embeddings and ensembles are helpful
- FT-Transformer is not justified
- Retrieval-augmented models are generally less performant

# Temporal shift

- GBDTs are less robust to temporal shift
- Realistic evaluation setups are important for healthy progress



# Summary

- A new benchmark with datasets, closer resembling real-world scenarios
- Sources: Kaggle and Yandex Eats, Maps, Weather, Lavka
- Datasets with 10M samples and feature-engineering *(with up-to 1000s of features)*
- All datasets have timestamps

# Outline

- Quick recap
- Tabular Benchmarks
- Pretraining in Tabular DL
- Cross-domain learning
- Generative tabular models
- Future directions

# Pretraining in DL: main idea

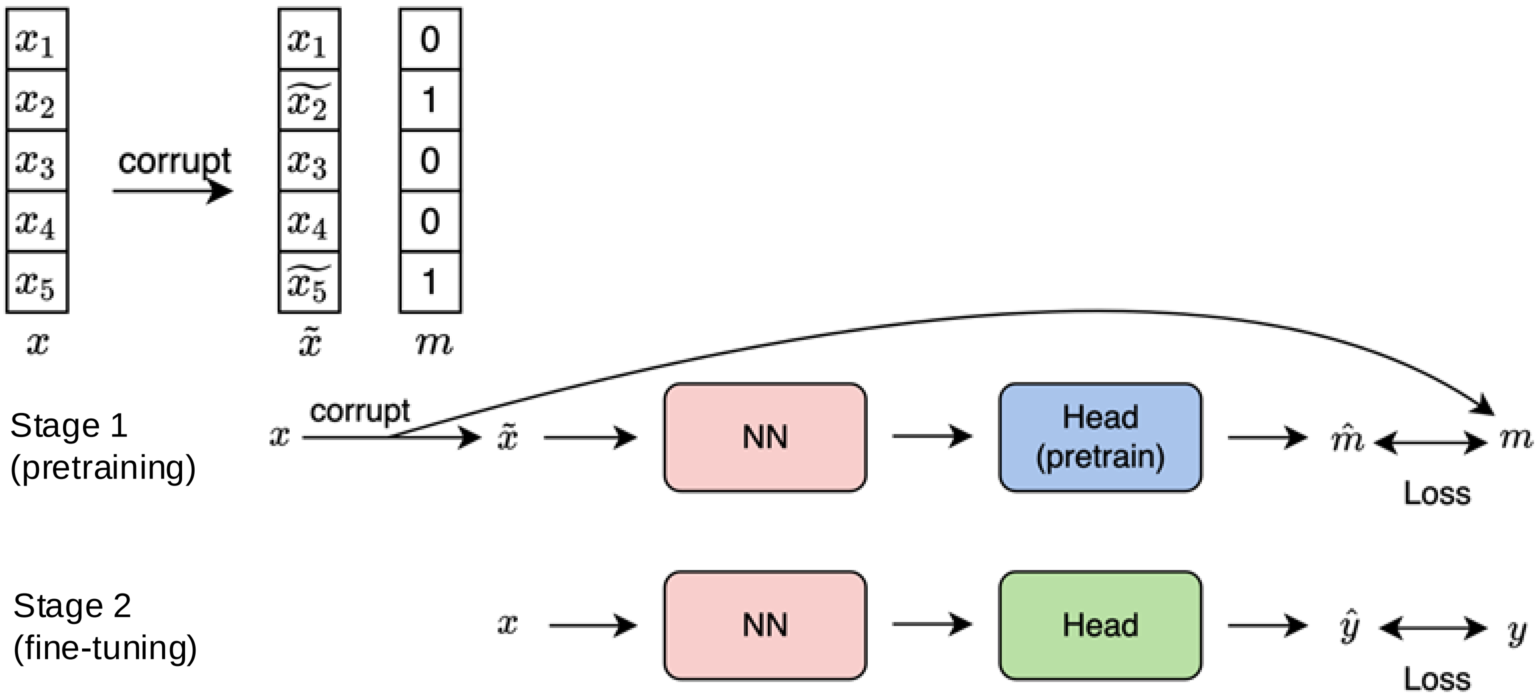
- To train the model to solve a related problem before the main learning process
    - Same data but dufferent tasks (e.g. with cheaper labels)
    - ”Extra” data from the same or a similar domain
- Inner logic of the pretrained model can be helpful for the target problem
- Provides better than random initialization for subsequent gradient optimization
- De facto standard for typical pipelines in NLP and CV
    - Contrastive learning
    - Self-prediction

# Pretraining in Tabular DL: specifics

- No ”extra” data
  - Need to pretrain on the main train set
- Lack of ”valid” data augmentations
  - Any augmentation can TODO the data distribution
  - Pretraining can be harmful
- Problems from a large number of domains
  - Need of the universal pretraining recipe

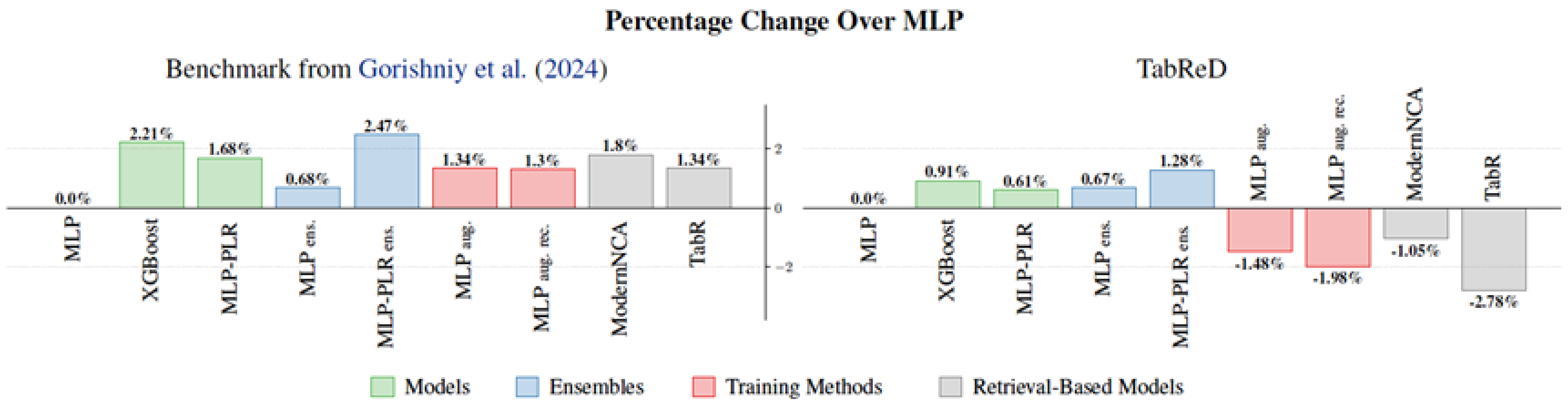
# Unsupervised pretraining for tabular data

**Mask prediction**



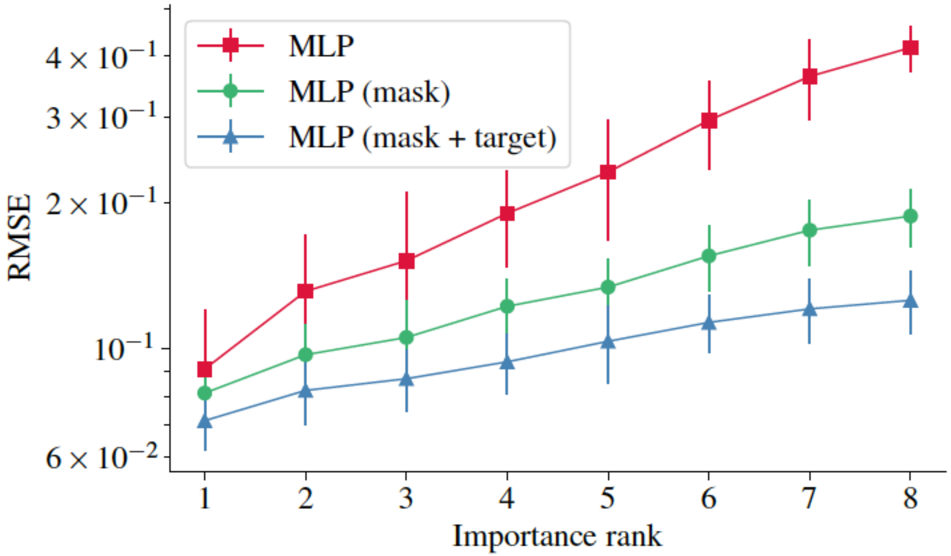

*Training*

# Experiments with pretraining




- All pretraining strategies perform on par to each other
- Pretraining is beneficial for both simple and advanced tabular DL models
- In temporal-shift based evaluation, pretraining can be harmful

# When and why pretraining helps?




- An experiment on synthetic data with controllable feature importances
- For different models, we measure the reconstruction quality of different features from the inner model representations
- Pretrained models capture less important (but still significant!) features better

# Conclusion

- Pretraining does have some potential in Tabular DL
- The choice of pretraining objective does not matter much
- The pretraining effect depends on the distribution shift between train and test
    - Effect is often negative when the shift is noticeable
    - The universal pretraining recipe is yet to discover

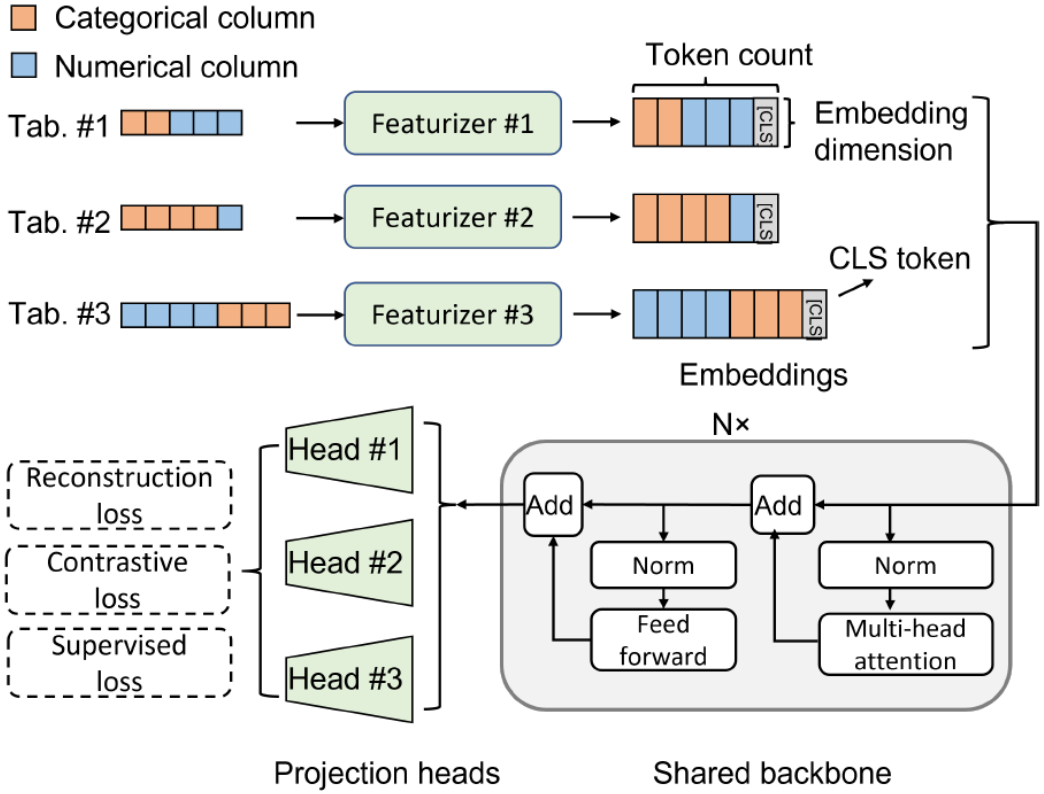
# Outline

- Quick recap
- Tabular Benchmarks
- Pretraining in Tabular DL
- Cross-domain learning
- Generative tabular models
- Future directions

# Main idea of cross-domain Tabular DL

- Leverage knowledge from one domain to improve predictions in another one
- Sounds like magic for tabular DL but …
- Sometimes does make sense (and even works)

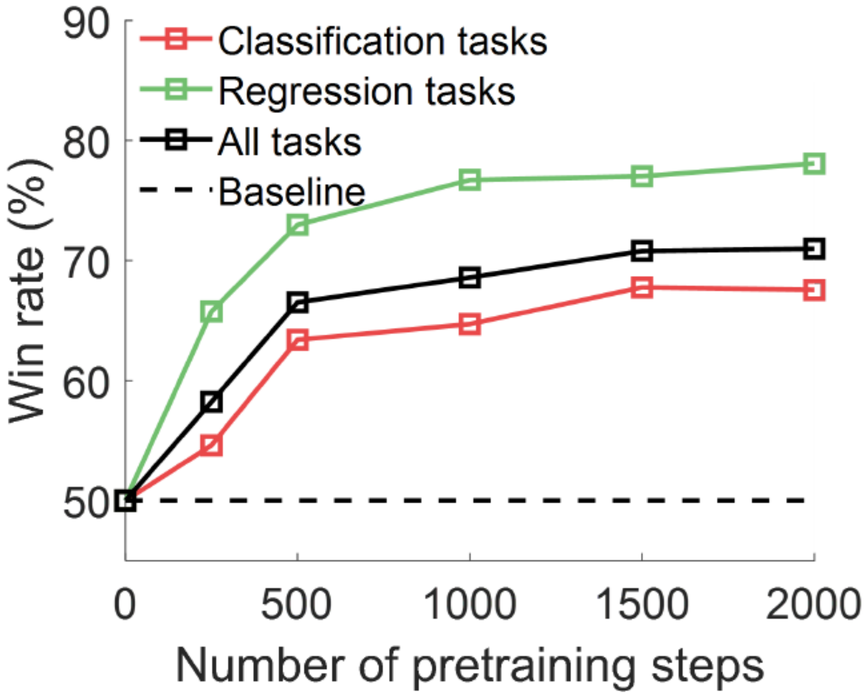
# XTAB (Zhu et al., ICML'2023)




- Pretrains a shared FT-Transformer backbone on many tabular tasks
- Feature tokenizers and final "heads" are not shared
- Can be used as a starting point for a new tabular task

Image credit: Zhu et al., ICML 2023

# XTAB: results



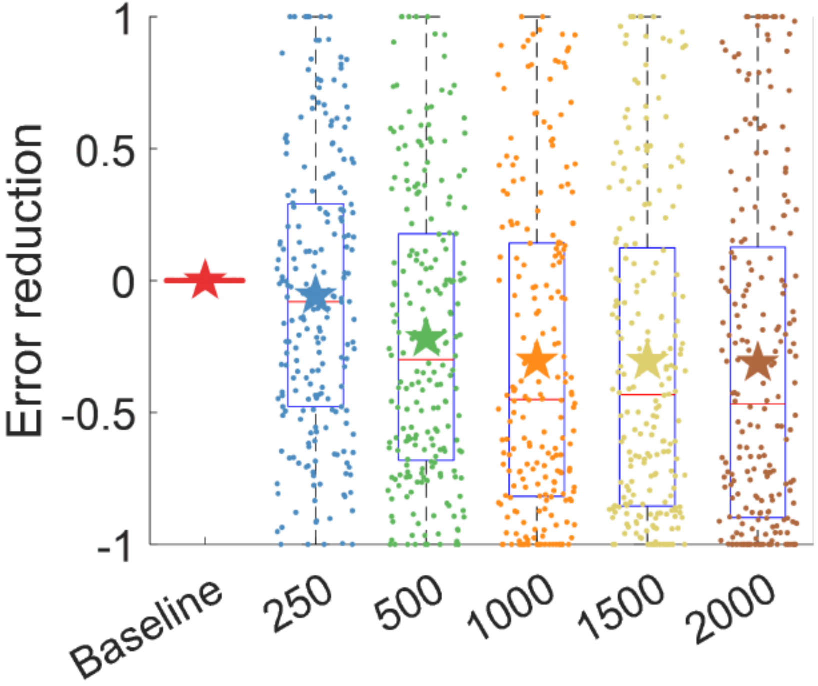
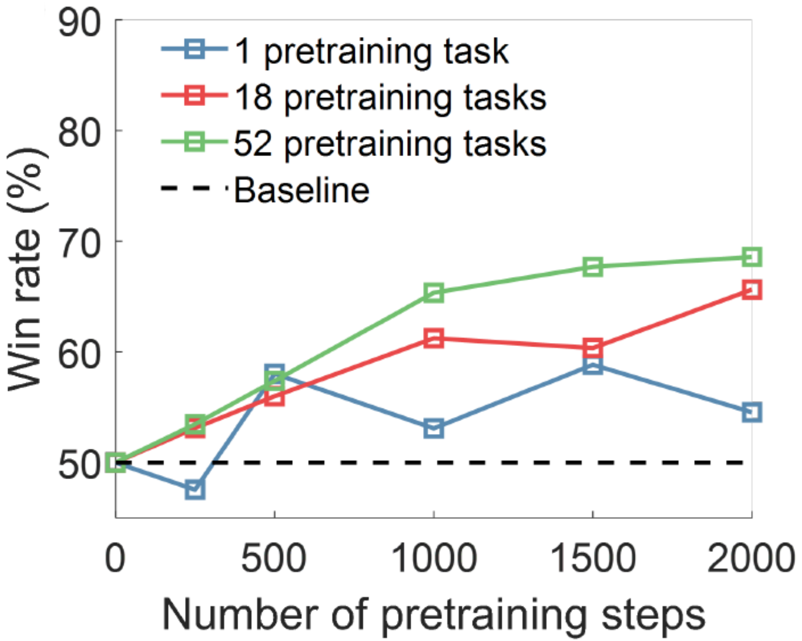




Image credit: Zhu et al., ICML 2023

# XTAB: dependence on the train size



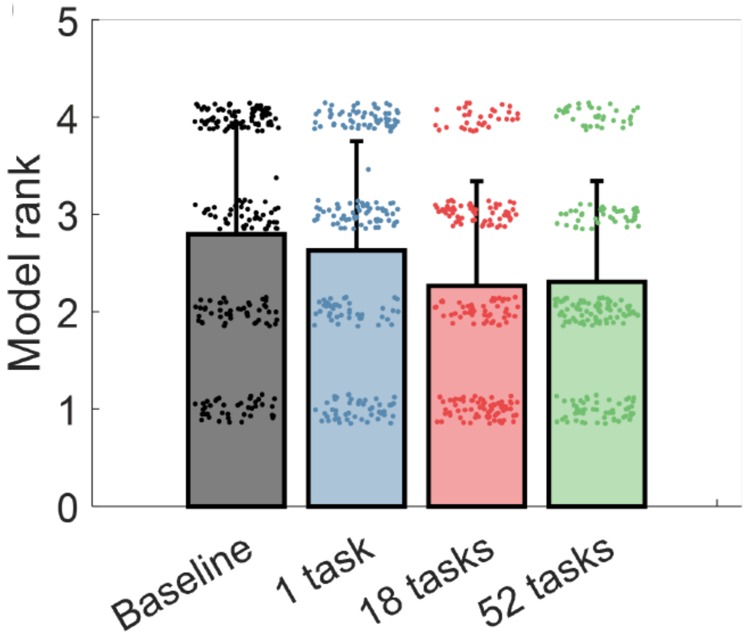
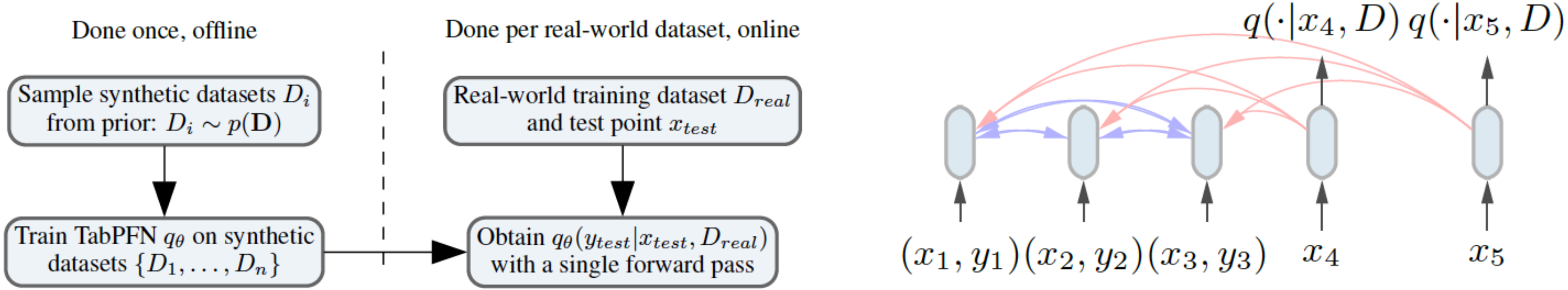




Image credit: Zhu et al., ICML 2023

# XTAB: conclusion

- Does provide some profit but …
- Is limited to Transformer-based architectures
  - Can be slow
  - Can be suboptimal
- Typical improvements are moderate

# TabPFN (Hollmann et al., ICLR'2023)



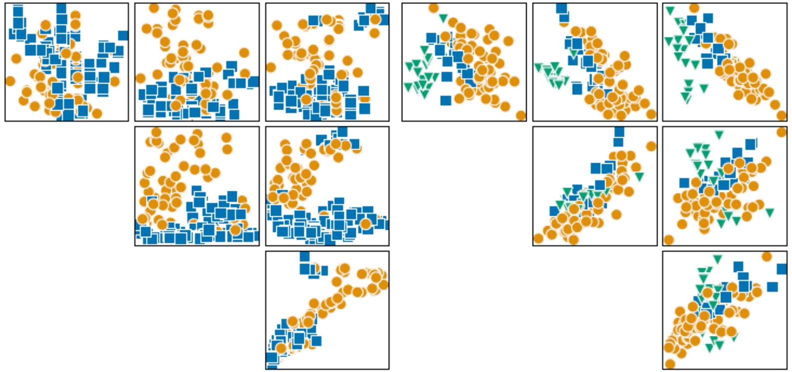

(a) Prior-fitting and inference

(b) Architecture and attention mechanism

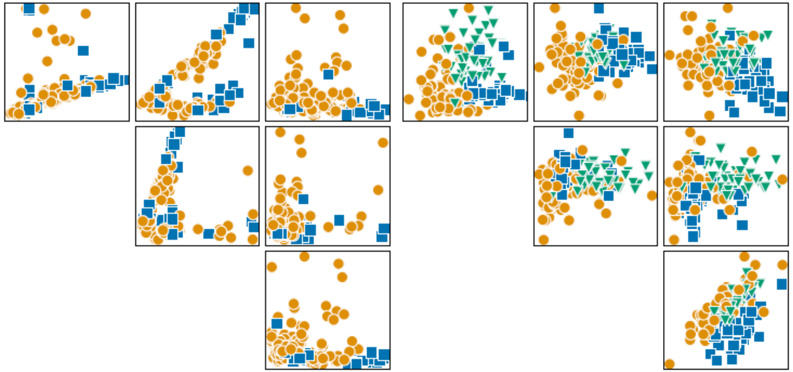
Image credit: Hollmann et al., ICLR 2023

# TabPFN: synthetics

- Synthetic datasets are sampled from an accurately designed prior



(a) Synthetic datasets



(b) Actual datasets

Image credit: Hollmann et al., ICLR 2023

# TabPFN: results

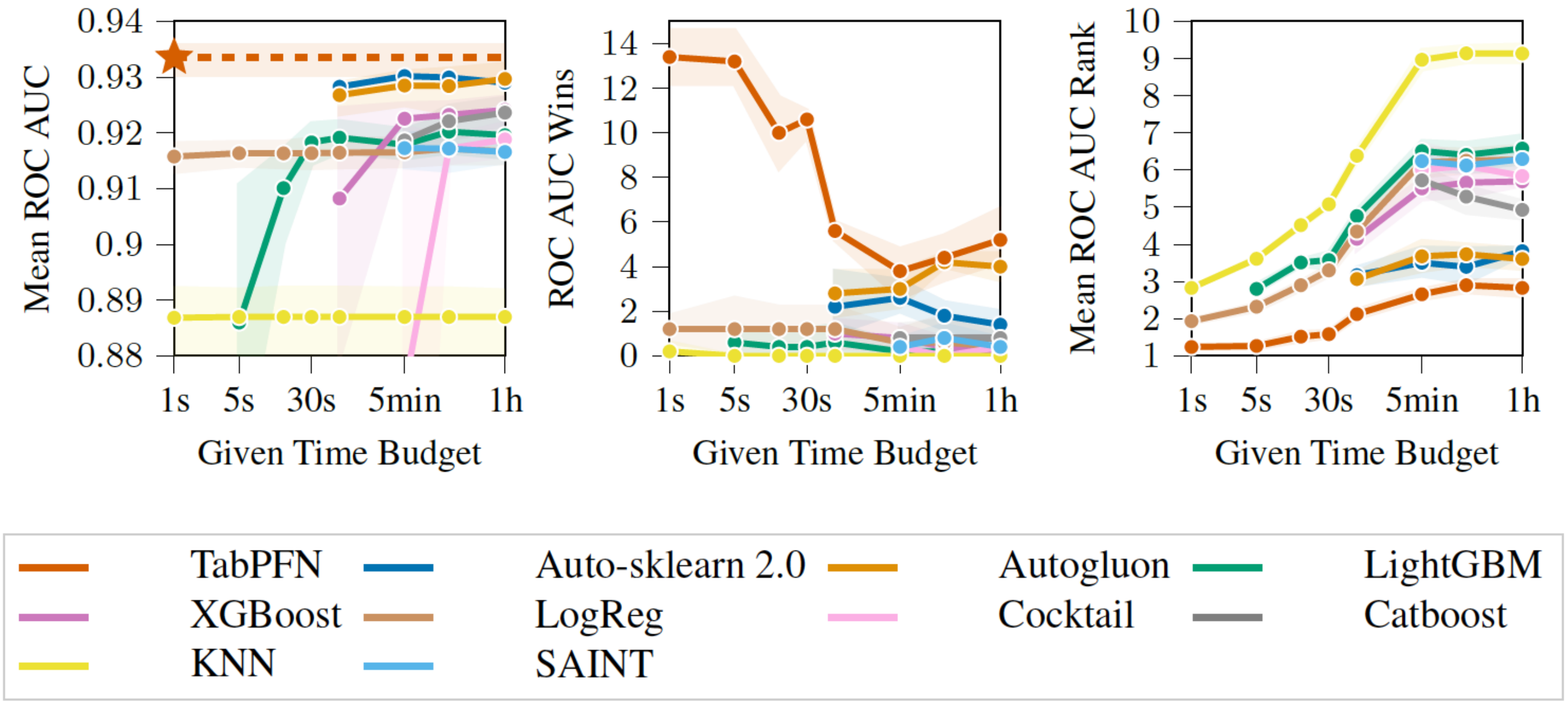
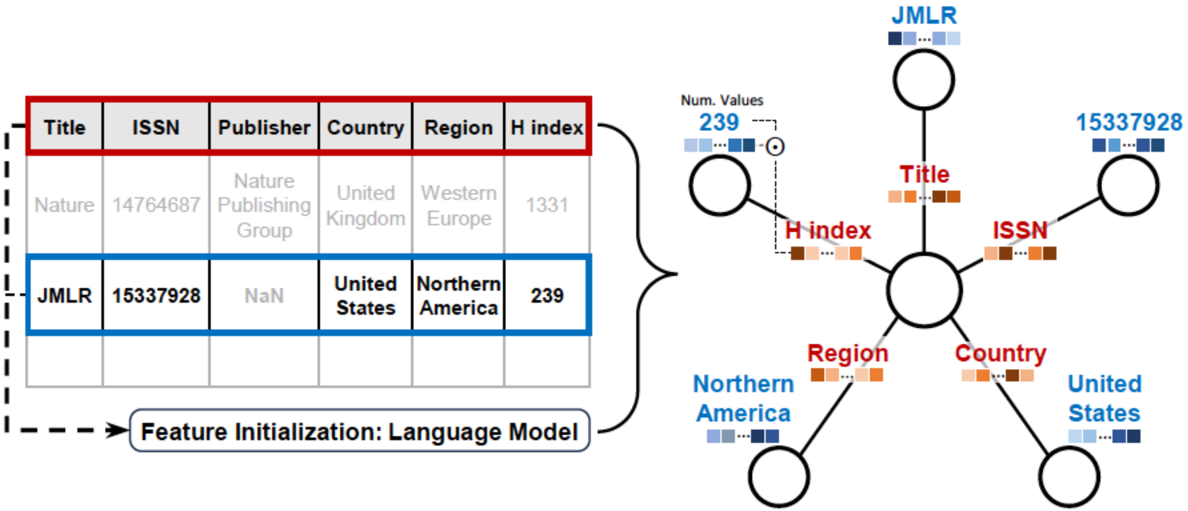



Image credit: Hollmann et al., ICLR 2023

# TabPFN: conclusion

- Very interesting and novel idea but …
- Is limited to Transformer-based architectures
- Is limited to small-scale problems
  - A lot of current research aims to scale TabPFN
- Focuses on a low-runtime-budget niche
  - In many applications, performance cannot be traded off against runtime

# CARTE (Kim et al., ICML'2024)




- Each datapoint is represented by a "star"-shaped graph
- "Textual" features are initialized based on LLM
- Special initialization of numerical features and the central node

Image credit: Kim et al., ICML 2024

# CARTE: pretraining from the external knowledge graph

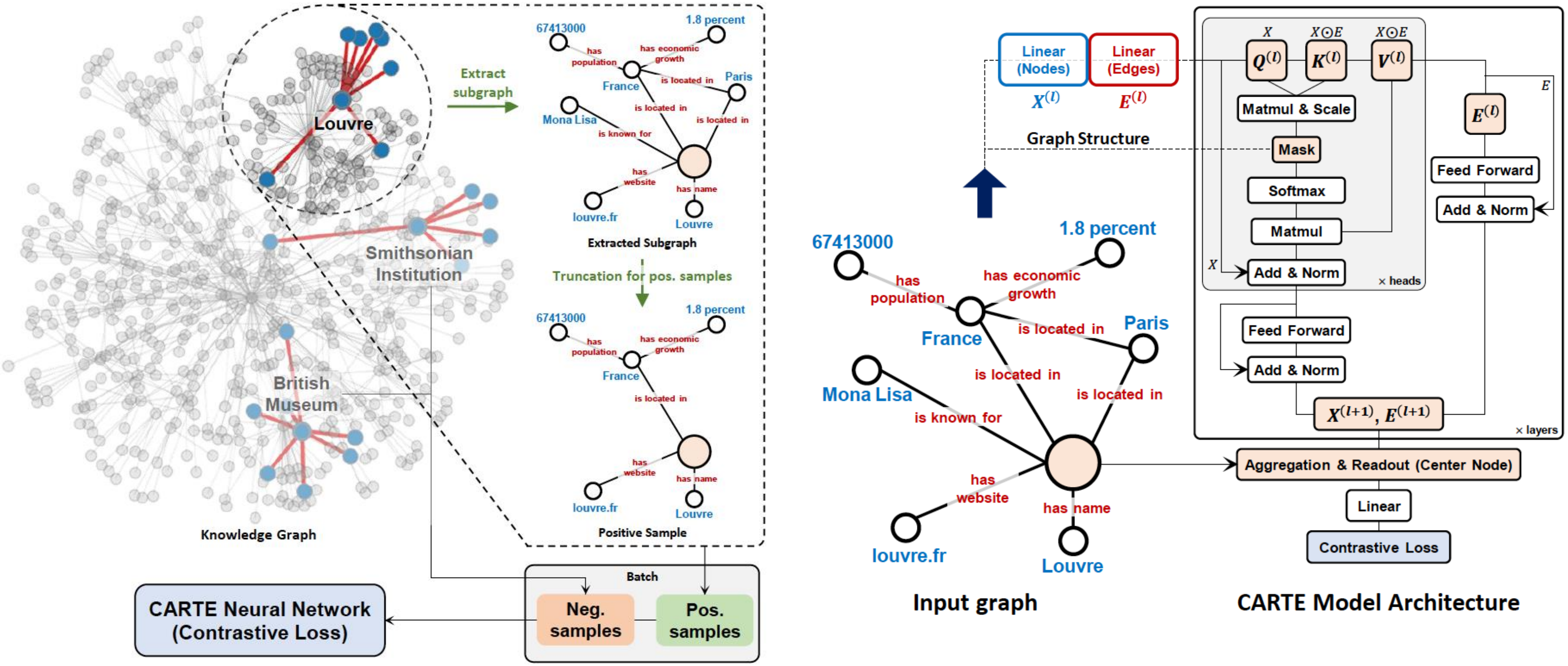



Image credit: Kim et al., ICML 2024

# CARTE: results

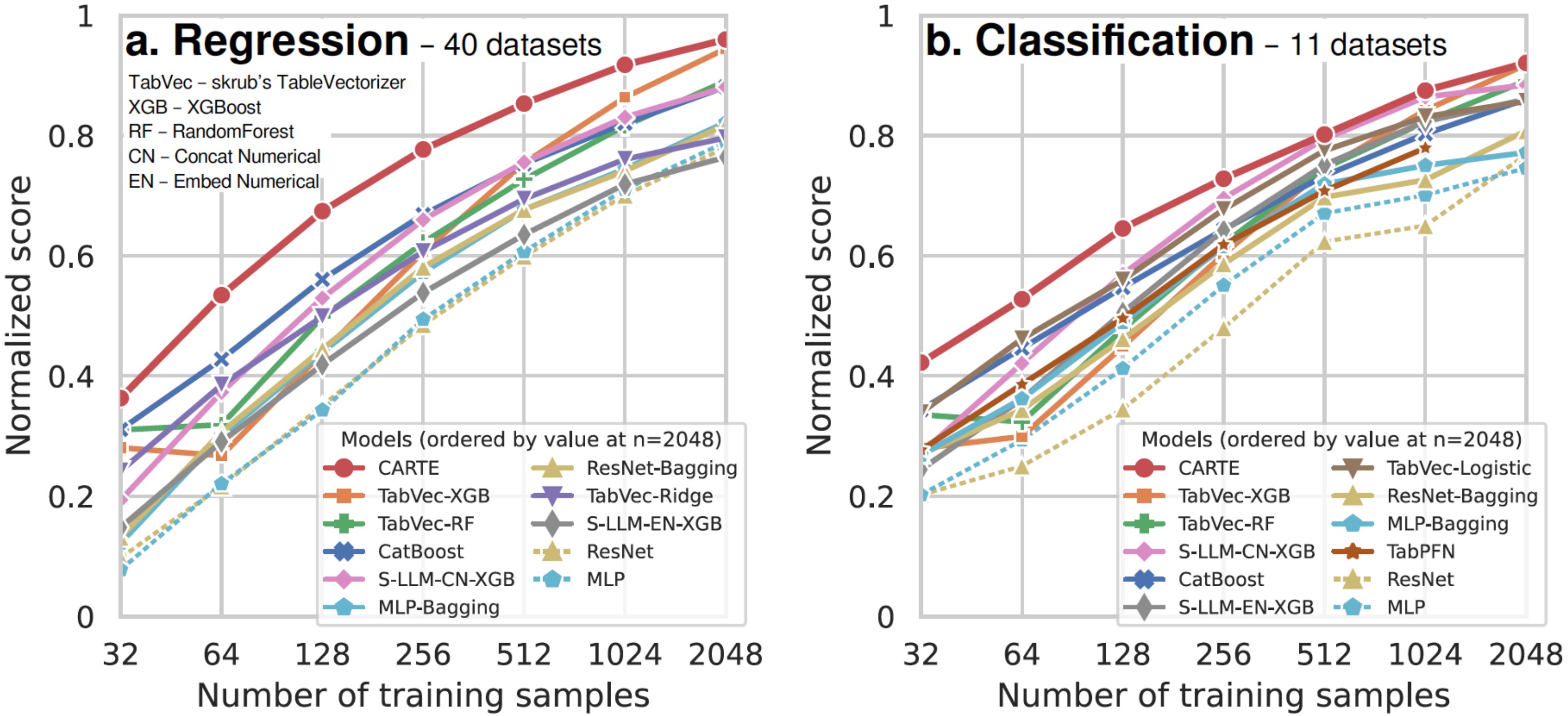



Image credit: Kim et al., ICML 2024

# CARTE: conclusion

- The method does work but …
- The success is shown only for Transformer-based architectures
- The success is shown only for small-scale problems (up to a few thousand objects)
- The method needs meaningful column names
- For certain domains there could be a lack of external knowledge graphs

# Outline

- Quick recap
- Tabular Benchmarks
- Pretraining in Tabular DL
- Cross-domain learning
- Generative tabular models
- Future directions

# Generative Modeling in ML

- Goal: to approximate the data distribution by a probabilistic model
- One of potential applications: to produce useful synthetic data
- Several families of methods exist: GAN, VAE, NF, DDPM

# Our work: TabDDPM

- Diffusion models were shown to outperform GAN/VAE/NF for images
- GAN/VAE were used for tabular data but without much success
- Let’s use diffusion models for tabular data!

# What are diffusion models?

- Forward process gradually adds noise to an initial sample with the predefined distributions $q(x_t|x_{t-1})$
- Reverse process gradually denoises a latent variable with distributions $p(x_{t-1}|x_t)$ that are approximated by a neural network
- For example, Gaussian distribution for continuous data and categorical distributions for categorical data

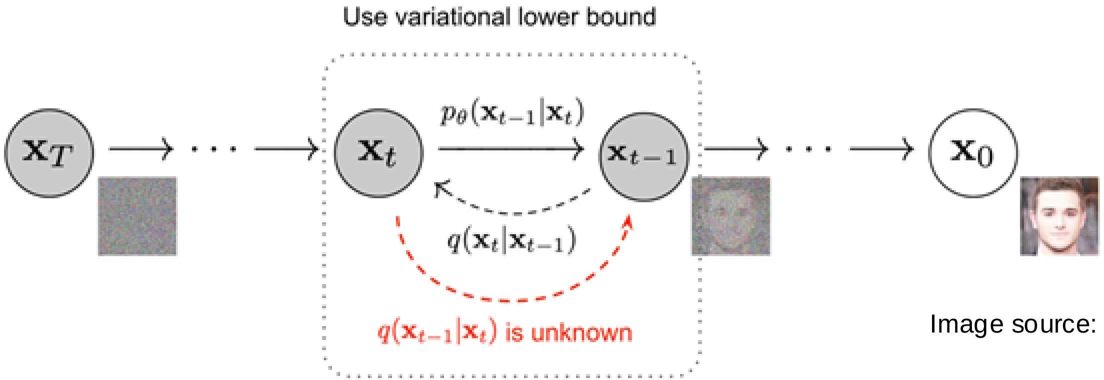
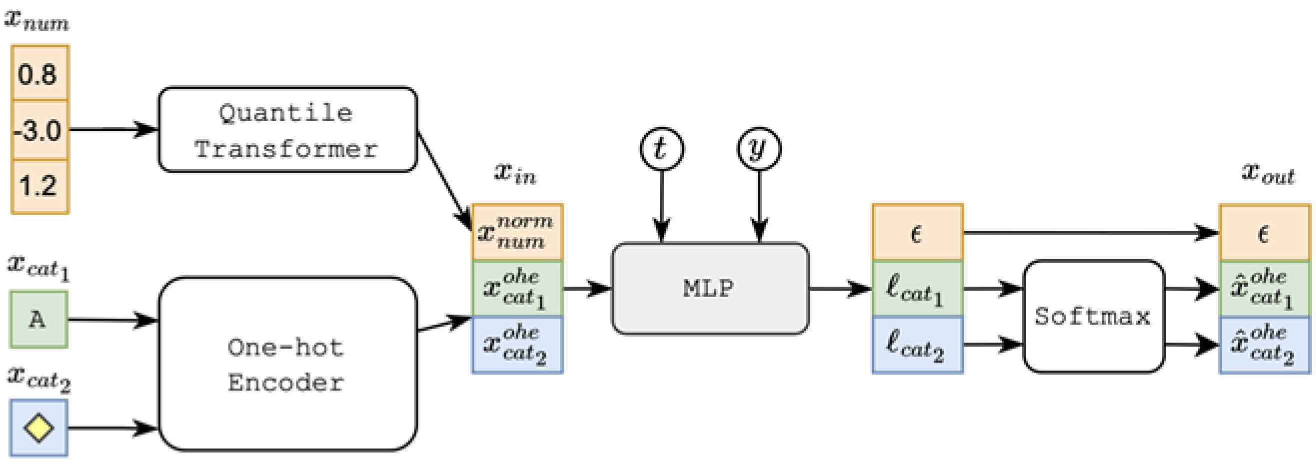


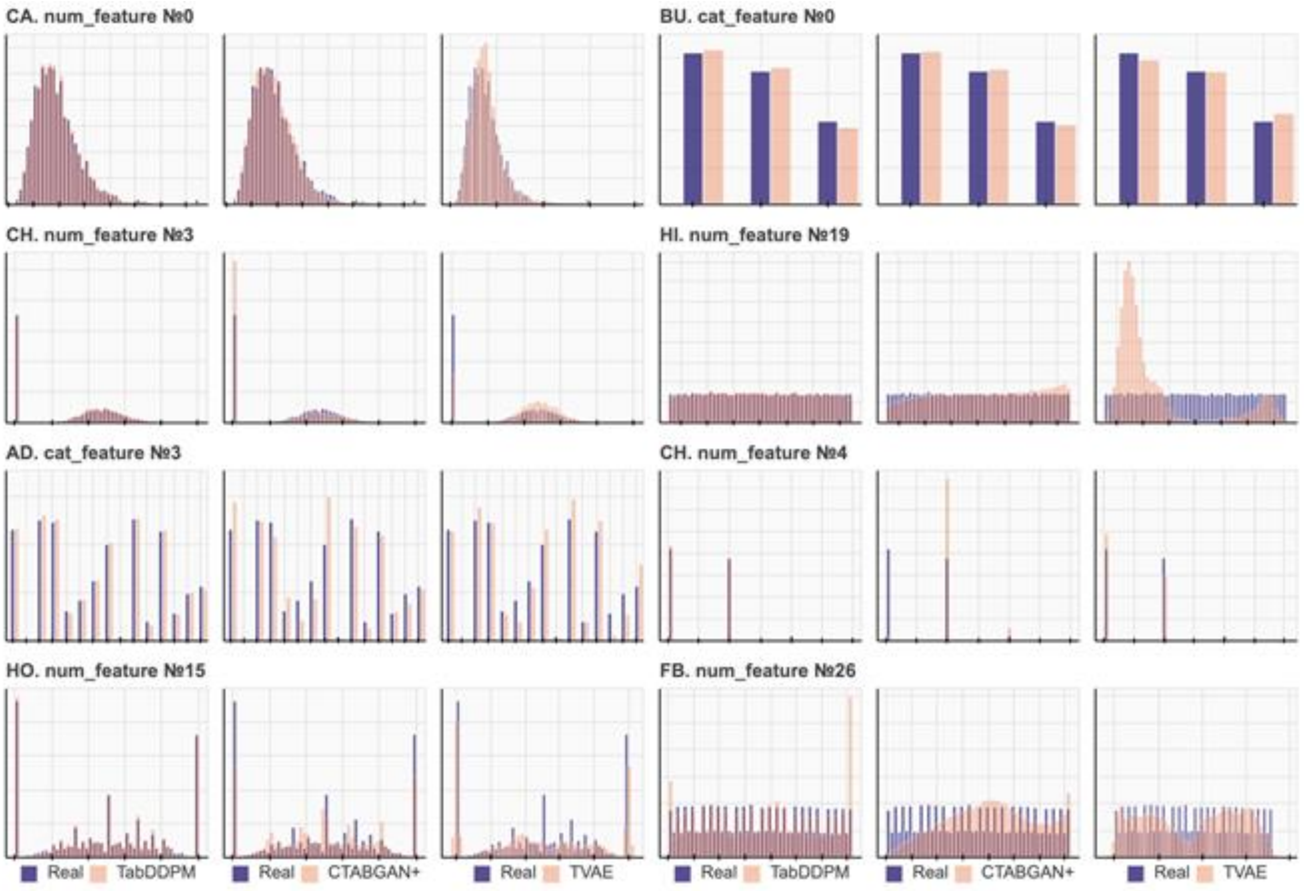

Image source: https://lilianweng.github.io

# TabDDPM

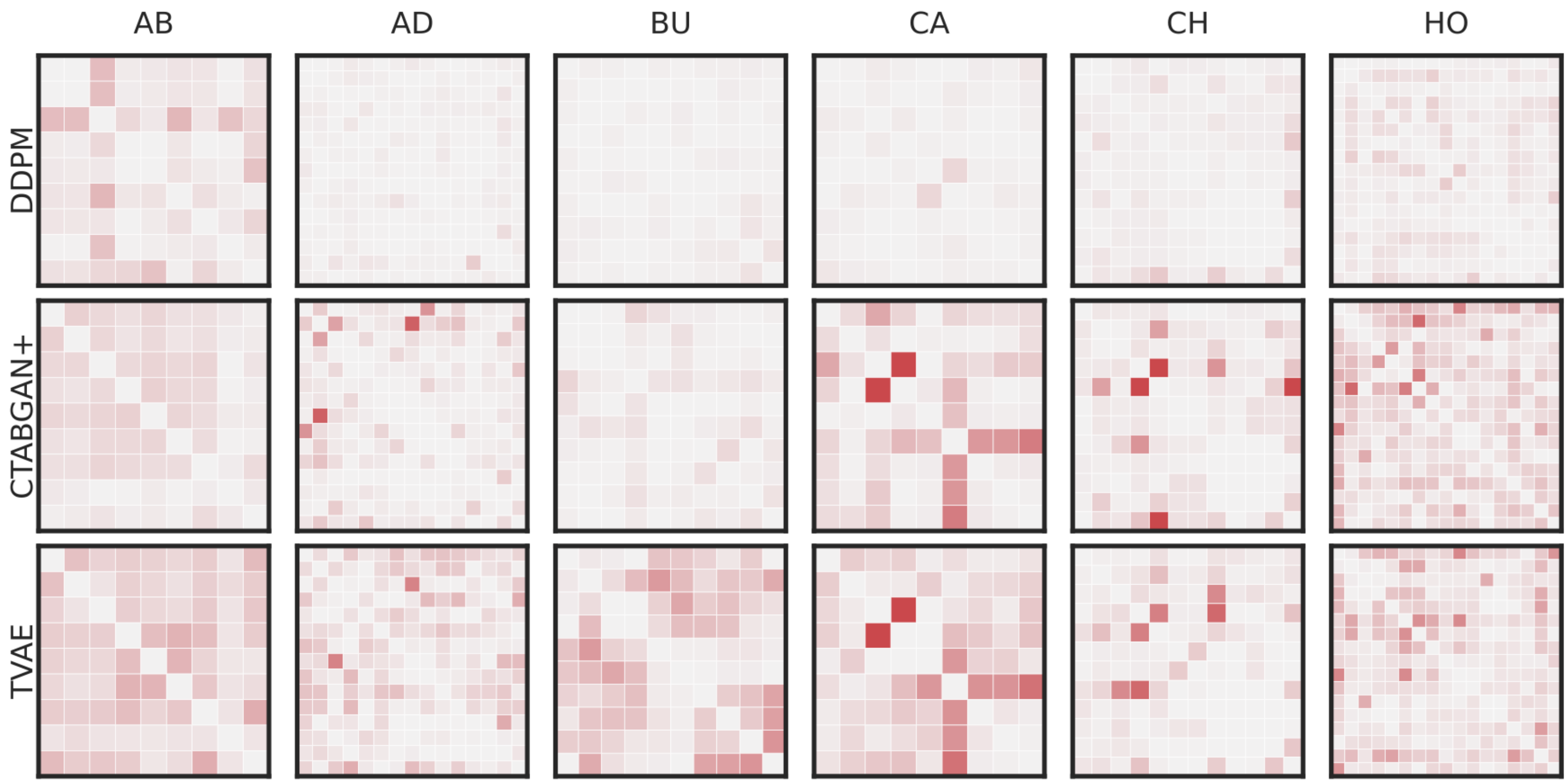
- Gaussian diffusion for numerical features
- Multinomial diffusion (Hoogeboom et al., 2021) for categorical features
- TabDDPM models joint distribution since MLP takes both numerical and categorical features to approximate reverse process
- Consider regression target as an additional feature
- Final loss is *sum* of gaussian DDPM and categorical DDPM losses



# Individual feature distributions

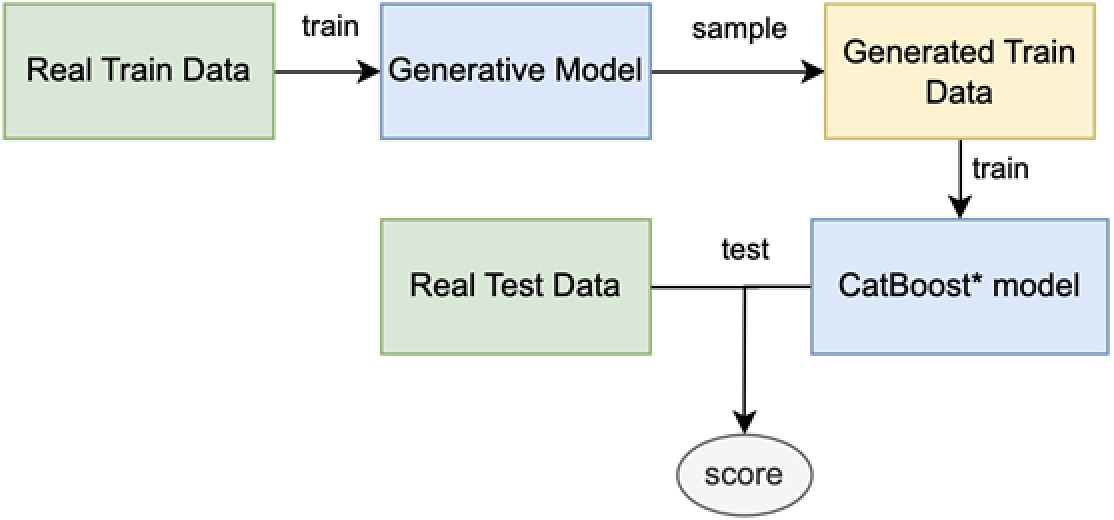


# Correlation matrices



# Evaluation

- Machine Learning utility (Xu et al., 2019)
- Privacy metrics




*(Prokhorenkova et al., 2018)

Compare this score with the real one

# ML utility with CatBoost model

Average rank (over 16 datasets) with std in terms of ML utility of synthetic data

1 – the best

5 – the worst

| Model | Avg. rank | Std of rank |
|---|---|---|
| CTGAN | 4.25 | 1.06 |
| TVAE | 3.81 | 0.83 |
| CTABGAN+ | 3.63 | 1.02 |
| SMOTE | 1.75 | 0.84 |
| TabDDPM | 1.56 | 0.60 |

SMOTE (Chawla et al., 2002) – linear interpolation of two random samples from train

# ML utility with CatBoost model

Average rank (over 16 datasets) with std in terms of ML utility of synthetic data

1 – the best

5 – the worst

| Model | Avg. rank | Std of rank |
|---|---|---|
| CTGAN | 4.25 | 1.06 |
| TVAE | 3.81 | 0.83 |
| CTABGAN+ | 3.63 | 1.02 |
| SMOTE | 1.75 | 0.84 |
| TabDDPM | 1.56 | 0.60 |

SMOTE (Chawla et al., 2002) – linear interpolation of two random samples from train

Main Conclusions:

- TabDDPM outperforms GAN/VAE-based baselines
- SMOTE is a simple and strong baseline

# ML utility with Catboost. Numbers.

- TabDDPM performs on par with SMOTE
- Real score is almost always the highest one

*Table 5.* The values of machine learning efficiency computed w.r.t. the state-of-the-art tuned CatBoost model.

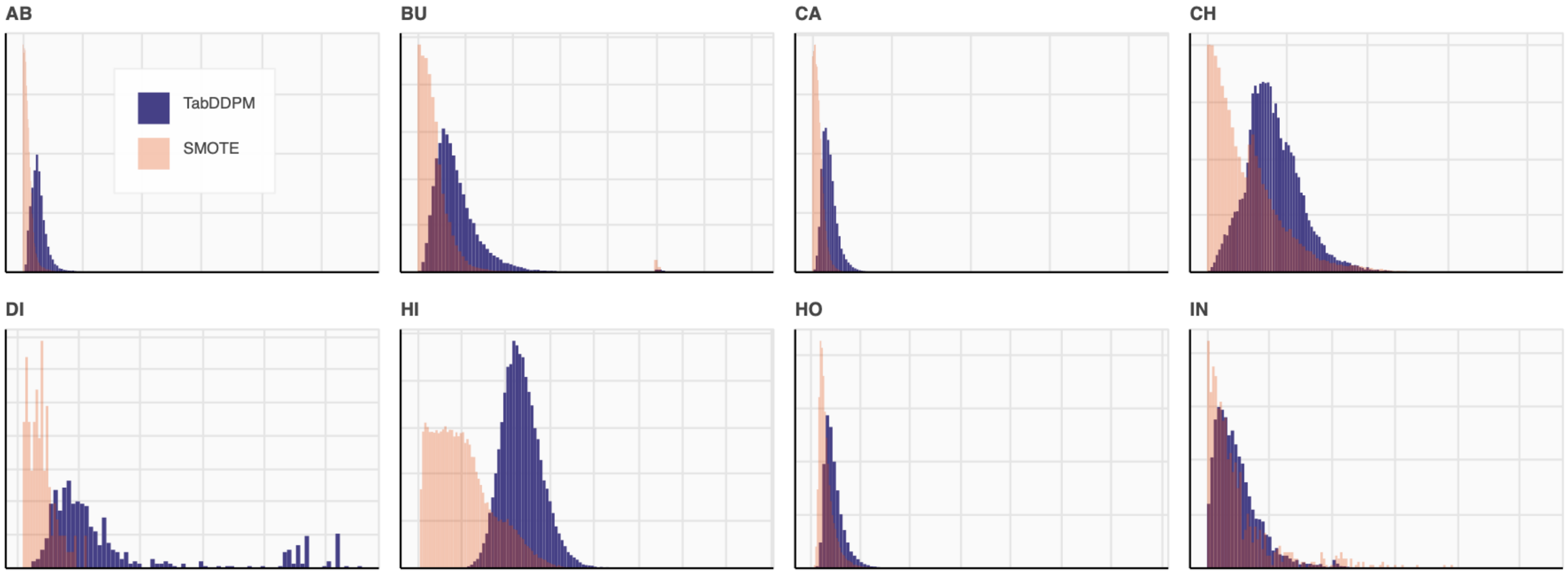
| | AB ($R2$) | AD ($F1$) | BU ($F1$) | CA ($R2$) | CAR ($F1$) | CH ($F1$) | DE ($F1$) | DI ($F1$) |
|---|---|---|---|---|---|---|---|---|
| CTGAN | $0.420_{\pm.004}$ | $0.789_{\pm.001}$ | $0.867_{\pm.003}$ | $0.686_{\pm.003}$ | $0.730_{\pm.001}$ | $0.723_{\pm.006}$ | $\mathbf{0.699_{\pm.002}}$ | $0.459_{\pm.096}$ |
| TVAE | $0.433_{\pm.008}$ | $0.781_{\pm.002}$ | $0.864_{\pm.005}$ | $0.752_{\pm.001}$ | $0.717_{\pm.001}$ | $0.732_{\pm.006}$ | $0.656_{\pm.007}$ | $\mathbf{0.714_{\pm.039}}$ |
| CTABGAN | – | $0.783_{\pm.002}$ | $0.855_{\pm.005}$ | – | $0.717_{\pm.001}$ | $0.688_{\pm.006}$ | $0.644_{\pm.011}$ | $\mathbf{0.731_{\pm.022}}$ |
| CTABGAN+ | $0.467_{\pm.004}$ | $0.772_{\pm.003}$ | $0.884_{\pm.005}$ | $0.525_{\pm.004}$ | $0.733_{\pm.001}$ | $0.702_{\pm.012}$ | $0.686_{\pm.004}$ | $\mathbf{0.734_{\pm.020}}$ |
| SMOTE | $\mathbf{0.549_{\pm.005}}$ | $0.791_{\pm.002}$ | $0.891_{\pm.003}$ | $\mathbf{0.840_{\pm.001}}$ | $0.732_{\pm.001}$ | $0.743_{\pm.005}$ | $0.693_{\pm.003}$ | $0.683_{\pm.037}$ |
| TabDDPM | $\mathbf{0.550_{\pm.010}}$ | $\mathbf{0.795_{\pm.001}}$ | $\mathbf{0.906_{\pm.003}}$ | $0.836_{\pm.002}$ | $\mathbf{0.737_{\pm.001}}$ | $\mathbf{0.755_{\pm.006}}$ | $0.691_{\pm.004}$ | $\mathbf{0.740_{\pm.020}}$ |
| Real | $0.556_{\pm.004}$ | $0.815_{\pm.002}$ | $0.906_{\pm.002}$ | $0.857_{\pm.001}$ | $0.738_{\pm.001}$ | $0.740_{\pm.009}$ | $0.688_{\pm.003}$ | $0.785_{\pm.013}$ |

| | FB ($R2$) | GE ($F1$) | HI ($F1$) | HO ($R2$) | IN ($R2$) | KI ($R2$) | MI ($F1$) | WI ($F1$) |
|---|---|---|---|---|---|---|---|---|
| CTGAN | $0.443_{\pm.005}$ | $0.333_{\pm.013}$ | $0.575_{\pm.006}$ | $0.433_{\pm.005}$ | $0.745_{\pm.009}$ | $0.772_{\pm.005}$ | $0.783_{\pm.005}$ | $0.749_{\pm.015}$ |
| TVAE | $0.685_{\pm.003}$ | $0.434_{\pm.006}$ | $0.638_{\pm.003}$ | $0.493_{\pm.006}$ | $0.784_{\pm.010}$ | $0.824_{\pm.003}$ | $0.912_{\pm.001}$ | $0.501_{\pm.012}$ |
| CTABGAN | – | $0.392_{\pm.006}$ | $0.575_{\pm.004}$ | – | – | – | $0.889_{\pm.002}$ | $\mathbf{0.906_{\pm.019}}$ |
| CTABGAN+ | $0.509_{\pm.011}$ | $0.406_{\pm.009}$ | $0.664_{\pm.002}$ | $0.504_{\pm.005}$ | $0.797_{\pm.005}$ | $0.444_{\pm.014}$ | $0.892_{\pm.002}$ | $0.798_{\pm.021}$ |
| SMOTE | $\mathbf{0.803_{\pm.002}}$ | $\mathbf{0.658_{\pm.007}}$ | $\mathbf{0.722_{\pm.001}}$ | $0.662_{\pm.004}$ | $\mathbf{0.812_{\pm.002}}$ | $\mathbf{0.842_{\pm.004}}$ | $0.932_{\pm.001}$ | $\mathbf{0.913_{\pm.007}}$ |
| TabDDPM | $0.713_{\pm.002}$ | $0.597_{\pm.006}$ | $\mathbf{0.722_{\pm.001}}$ | $\mathbf{0.677_{\pm.010}}$ | $0.809_{\pm.002}$ | $\mathbf{0.833_{\pm.014}}$ | $\mathbf{0.936_{\pm.001}}$ | $\mathbf{0.904_{\pm.009}}$ |
| Real | $0.837_{\pm.001}$ | $0.636_{\pm.007}$ | $0.724_{\pm.001}$ | $0.662_{\pm.003}$ | $0.814_{\pm.001}$ | $0.907_{\pm.002}$ | $0.934_{\pm.000}$ | $0.898_{\pm.006}$ |

# Privacy. Distance to closest record (DCR)

- For each synthetic sample, we find the minimum distance to real datapoints and take the mean of these distances
- <u>Low</u> DCR values = all synthetic samples are essentially copies of some real datapoints
- <u>Larger</u> DCR values = generative model can produce something “new” rather than just copies of real data

# Histograms of DCR values for TabDDPM and SMOTE



# DCR comparison

Average rank (over 16 datasets) with std in terms of DCR

1 – the best

4 – the worst

| Model | Avg. rank | Std of rank |
|---|---|---|
| TVAE | 2.31 | 0.95 |
| CTABGAN+ | 1.56 | 0.81 |
| SMOTE | 3.44 | 1.09 |
| TabDDPM | 2.69 | 0.79 |

Main Conclusions:

- TabDDPM outperforms SMOTE
- GAN/VAE methods show high DCR but generate useless (in terms of ML utility) samples

# Conclusion

- Diffusion models generate tabular data of higher quality than GAN/VAE data
  - But still not enough for usage as ”useful” synthetics
- “Old-school” SMOTE is a strong baseline that should not be overlooked
- TabDDPM is a step forward towards strong yet private method

# Outline

- Quick recap
- Tabular Benchmarks
- Pretraining in Tabular DL
- Cross-domain learning
- Generative tabular models
- Future directions

# Future of Tabular DL research

- More theory and understanding
  - Optimization dynamics
  - Dealing with ‘high-frequencies’
- Synergy with Graph ML and GNNs
  - For graphs with tabular features in the nodes/edges
  - For multi-table problems with relations between tables
- Exploit LLM for Tabular problems
  - Use textual metadata about features
  - Multi-modal datasets
- Usability
  - Tooling

Questions?