TabNet: A Deep Learning Breakthrough for Tabular Data

12 min read · Feb 24, 2025

Dong-Keon Kim

How TabNet Bridges the Gap Between Neural Networks and Gradient Boosting Trees

Introduction

About

This study introduces **TabNet**, a deep learning model optimized for **tabular data processing.** TabNet is an **end-to-end neural network** designed to directly handle raw tabular data, leveraging an **attention mechanism** to selectively interpret each feature. The authors demonstrate that TabNet offers **interpretability**, **tunability through multiple hyper-parameters**, and **applicability to complex tasks** such as **self-supervised learning.**

Background

Most **deep learning models** have been primarily developed to handle structured data types such as **images**, **text**, and **audio**, achieving **remarkable performance** in these domains. However, **deep learning approaches for tabular data** remain relatively underexplored in comparison.

Traditionally, **decision tree-based machine learning models**, such as **XGBoost** and **LightGBM**, have been the standard for tabular data tasks. In contrast, **deep learning models for tabular data** have been largely limited to **Multilayer Perceptrons (MLPs)**, as more sophisticated architectures like **CNNs** and **LSTMs** are primarily designed for **high-dimensional image and text processing.**

Despite these challenges, the authors highlight several **advantages** of using deep learning for tabular data:

1. **Efficiently encoding multiple data types**, including tabular data alongside images.
2. **Reducing the need for extensive feature engineering**, a key requirement in traditional tree-based models.
3. **Enabling learning from streaming data**, making models more adaptable to real-time applications.
4. **Facilitating representation learning in an end-to-end manner**, unlocking valuable applications such as **data-efficient domain adaptation.**

Method

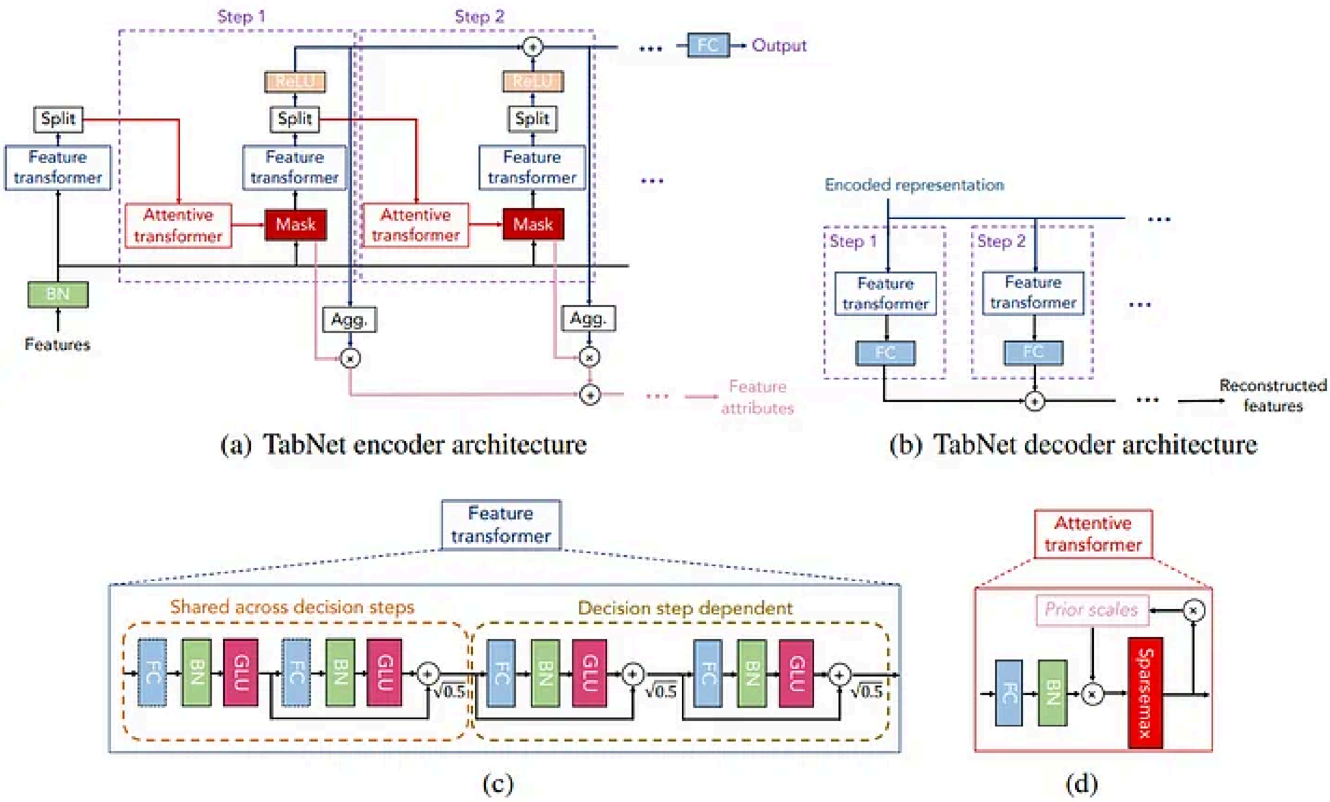


Fig 1. Overall structure of TabNet. (a) is the encoder part that encodes the input data with the transformer manner. (b) indicates a decoder that restores the encoded representation to the original data representation. And (c) and (d) show the structure of the feature transformer and the attentive transformer, respectively. (Source)

The TabNet proposed in this paper learns data through several steps. Similar to the existing decision tree mechanism, each step in **TabNet selects features among the given inputs for reducing dimension**. Here, an **attention manner** is utilized in each process of selecting features. The authors mention that these elements of deep learning at each step improve the learning capacity of the model.

The authors consider both numeric and categorical types of tabular data. As in other deep learning studies, input categorical data is preprocessed through trainable embedding while the raw numerical features are inserted to the model directly. Instead of global feature normalization for the input data, a batch normalization is applied.

The feature $\mathbf{f} \in \Re^{B \times D}$ is the hidden features of each decision steps, where B is the batch size and D means the dimension of the input data. TabNet outputs the result after going through a total of N_{steps}. The authors propose this method by referring to the architecture that processing visual and text data. According to the overall flow in **Fig 1**, the structure of the presented TabNet is similar to recursive-based neural network models.

Feature Selection

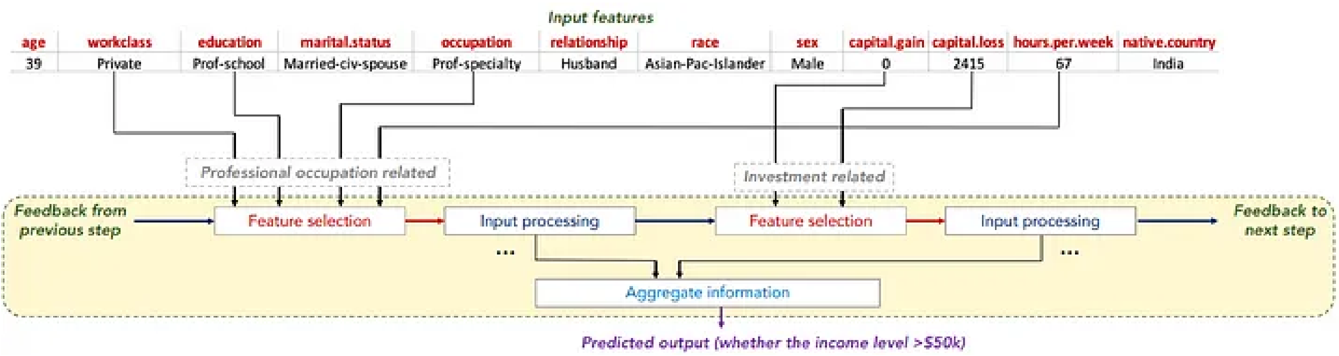


Fig 2. TabNet's sparse feature selection exemplified for Adult Census Income prediction. (Source)

The authors design a concept of the model that selects significant features. **Fig 2.** describes that TabNet does not use all features for model training. To select features, a learnable mask $\mathbf{M[i]} \in \Re^{B \times D}$ is considered. Note that **i** indicates i-th step in the encoder. The sum of the elements of this mask matrix is **1**, which means that the importance is evaluated relatively. Like the description in **Fig 3.**, The mask selects the sparse salient features which makes each step not to flow irrelevantly of the task.

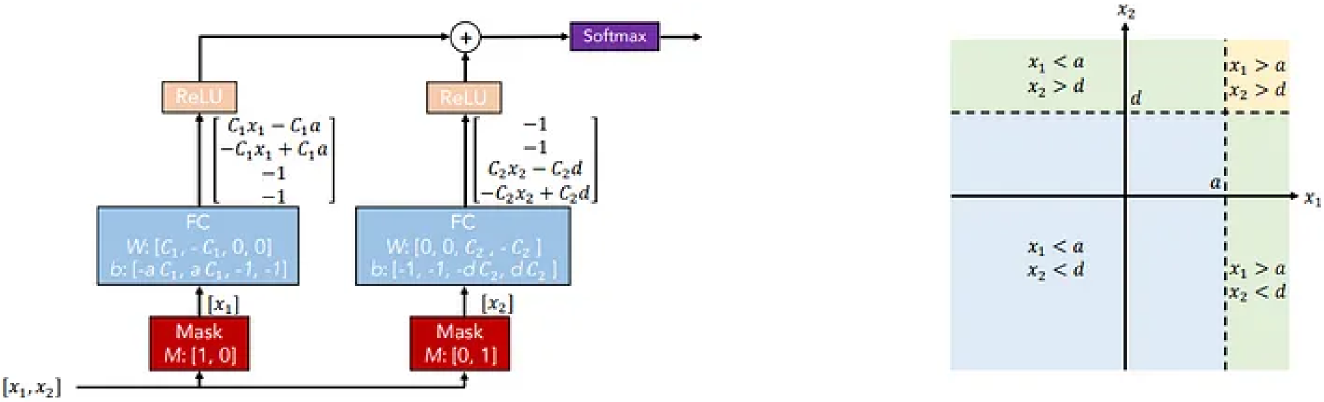


Fig 3. Illustration of classification using DNN (left) and the corresponding decision manifold (right). With trained masks, the values of specific features are fixed. Here, the C_1 and C_2 are related to the sharpness of the decision boundary. (Source)

An attentive transformer is employed to obtain the masks using the output features from the preceding step. The authors express how to obtain the current-phase mask from the results of the previous step as follows.

$$\mathbf{M}[\mathbf{i}] = \mathrm{sparsemax}(\mathbf{P}[\mathbf{i}-\mathbf{1}] \cdot \mathrm{h}_i(\mathbf{a}[\mathbf{i}-\mathbf{1}]))$$

Here, **P[i - 1]** is the *prior scale* term which denotes how much a specific parameter has been used previously. This can be written like the equation below.

$$\mathbf{P}[\mathbf{i}] = \prod_{j=1}^{i} (\gamma - \mathbf{M}[\mathbf{j}])$$

The prior scale is tunable by relaxation parameter γ. As the value of this variable increases, features can be selected more flexibly at each step. This is similar to the forget gate of LSTM. They are the same in that it controls how much information amount determined in the previous step is reflected in the next step. In particular, the prior term of the first step **P[0]** is set to $\mathbf{1}^{B \times D}$ because there is no element of the previous step. This way is also modifiable to made some elements to **0** (some columns of the input data are unused) to help the training phase in a self-supervised learning task.

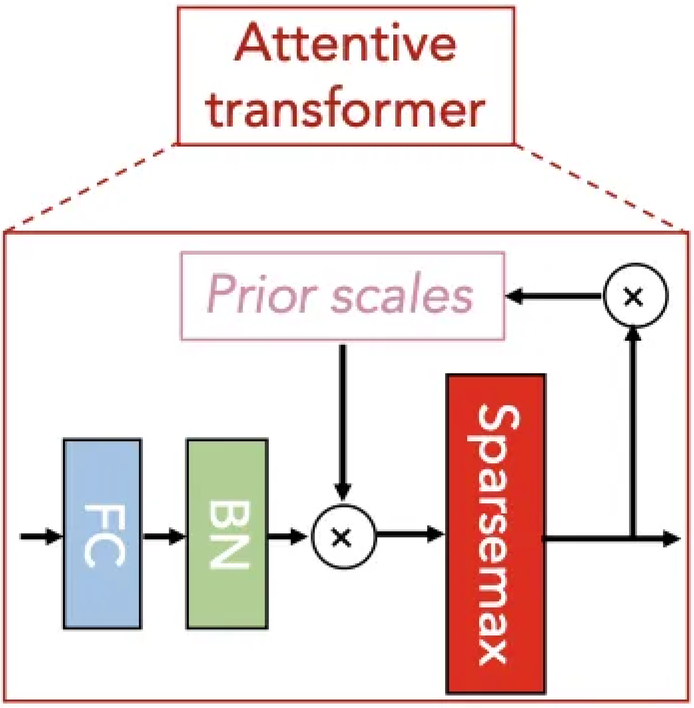


Fig 4. Structure of the attentive transformer. This is a enlarged picture of Fig 1. (d). (Source)

With the trainable function h_i, the preceding outcomes are processed. As described in **Fig 4**, h consists of a fully connected (FC) neural network and a following batch normalization (BN) layer. The feature obtained through h_i is multiplied by the prior scale **P[i - 1]**.

Finally, the result is refined through *sparsemax* normalization. This method functions **to better select sparse but well salient features.** In tabular data, sometimes only a few columns have significant features while others are trivial. The authors describe the sparcity regularization in the following form of entropy.

$$L_{sparse} = \sum_{i=1}^{N_{steps}} \sum_{b=1}^{B} \sum_{j=1}^{D} \frac{-\mathbf{M}_{b,j}[\mathbf{i}]\log(\mathbf{M}_{b,j}[\mathbf{i}] + \epsilon)}{N_{steps \cdot B}}$$

The loss is used to control the sparsity of the selected feature. The constant \epsilon is a very small number for numerical stability. This is multiplied by coefficient γ_{sparse} and added to the overall loss. **This method is utilized to emphasize features in tabular data in which only a small number of columns show decisive features.**

Feature Processing

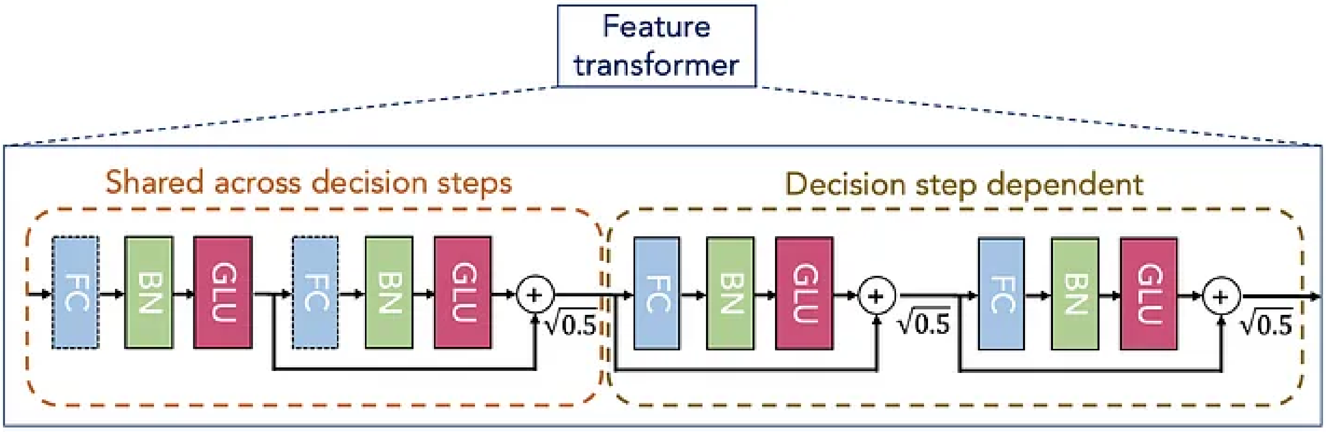


Fig 5. Architecture of the feature transformer. This is a enlarged picture of Fig 1. (c). (Source)

To further learn the selected features obtained through the mask, a feature transformer is devised. The additional learning model is called a feature transformer, and is denoted by f_i in this paper. With the trained mask **M[i]** from the feature selection manner and the hidden feature **f**, the output of the feature transformer is as follows.

$$[\mathbf{d}[\mathbf{i}], \mathbf{a}[\mathbf{i}]] = \mathrm{f}_i(\mathbf{M}[\mathbf{i}] \cdot \mathbf{f})$$

Note that the result of feature selection split in two. The first term **d[i]** $\in \Re^{B \times N_d}$ means decision step output and **a[i]** $\in \Re^{B \times N_a}$ stands for the prior scale for the next step. As shown in Fig 1, the result going out to the **ReLU** cell is **d[i]**, and the result going out to the attentive transformer of the next step is **a[i]**. For robust learning and parameter efficiency, the final prediction is output by accumulating the results of all steps with the adding manner.

The feature transformer structure is a simple MLP(Multi-Layer Perceptron) as shown in **Fig 5**. MLP consists of FC layer, BN layer and an activation function(GLU). Especially, there are residual connections between the MLP blocks. Especially, a ghost BN form is employed for preserving low-variance averaging among the batches. In this paper, the size of a virtual batch is expressed as B_v, and the corresponding momentum is called m_B. The authors additionally applied normalization with √0.5 right before the connection. Normalization stands for the stabilized learning throughout the model layer.

The hidden feature **a[i]** which is for the prior scales of the next step is returned in the middle of feature transformer, as expressed by the red dashed box in **Fig 4**. This output is input to the next attentive transformer. On the other hand, The hidden feature for decision output **d[i]** returned from the whole procedures in the feature transformer. The final decision output is expressed as **W(*final*)d**$_{out}$, where **W(*final*)** is a linear mapping, and **d**$_{out}$ is the **ReLU** sum of the **d[i]** in all steps.

Interpretability

The trained mask **M[i] is a kind of indicator that presents which features considered more important.** The authors show by way of example that the jth feature of the bth sample means no contribution to the decision when **M[i] = 0. Note that the masks of each step each can represent this feature importance.** Here, a coefficient that can weigh the relative importance of each step in the decision is required for combining the masks at different steps. The authors proposed the aggregated decision contribution of ith step of the bth sample with the equation below.

$$\eta_{\mathbf{b}}[\mathbf{i}] = \sum_{c=1}^{N_d} \mathrm{ReLU}(\mathbf{d}_{\mathbf{b},\mathbf{c}}[\mathbf{i}])$$

The larger this value, the greater the contribution to the overall mask combination.

With this metric, the author ultimately aims to see the importance of the input data. To do so, the feature importance mask is defined as follows.

$$\mathbf{M}_{\mathbf{agg-b,j}} = \frac{\sum_{i=1}^{N_{steps}} \eta_{\mathbf{b}}[\mathbf{i}]\mathbf{M}_{\mathbf{b,j}}[\mathbf{i}]}{\sum_{j=1}^{D}\sum_{i=1}^{N_{steps}} \eta_{\mathbf{b}}[\mathbf{i}]\mathbf{M}_{\mathbf{b,j}}[\mathbf{i}]}$$

Note that this property **M** is normalized so that it sums to **1** for all j. **Through this, it is possible to interpret the prediction result.**

Tabular self-supervised learning

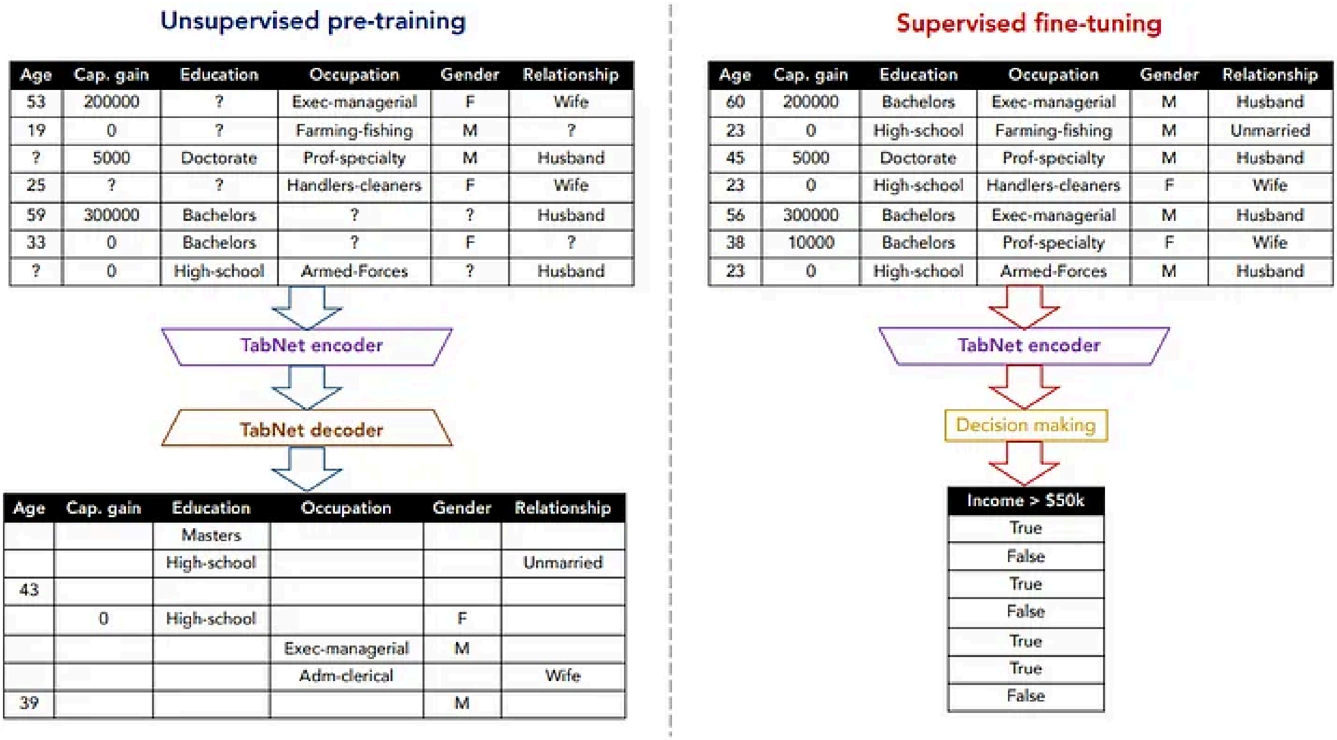


Fig 6. Description of self-supervised tabular learning. In this paper, the authors intend to use TabNet for the task of exploring unknown data as in the case on the left. (Source)

TabNet reproduces the result encoded by the previous process through a decoder. As shown in **Fig 1 (b)**, the decoder consists of stepwise feature transformers with following FC layers. The outputs are summed to obtain the reconstructed features.

The authors additionally propose **the usage for the self-supervised learning of this model**. As an example like **Fig 6.**, the authors introduce **the task of inferring missing values of input data.** Considering the binary mask **S** ∈ $\{0, 1\}^{B \times D}$, the TabNet is set the encoder to take **P[0]** ·**f** as input and the decoder to take **S**·**f** as output, where **P[0]** is initialized to (**1 — S**). Through this initialization, the **known features are further emphasized**, and **the missed feature is inferred the by multiplying the model result f and S.**

Reconstruction loss is defined as

$$\sum_{b=1}^{B}\sum_{j=1}^{D}\left|\frac{(\hat{\mathbf{f}}_{\mathbf{b,j}} - \mathbf{f}_{\mathbf{b,j}}) \cdot \mathbf{S}_{\mathbf{b,j}}}{\sqrt{\sum_{b=1}^{B}(\mathbf{f}_{\mathbf{b,j}} - 1/B\sum_{b=1}^{B}\mathbf{f}_{\mathbf{b,j}})^2}}\right|^2$$

Normalization adjusts different scales of features. The binary mask **S**(b, j)is independently sampled through the Bernoulli distribution at each iteration.

Experiment

The authors conducted very extensive experiments with the proposed model, including regression and classification. For the purposes of TabNet, no preprocessing was done for all datasets except for trainable scalar embedding of categorical data.

The loss functions are simply **cross-entropy** (classification) and **mean squared error** (regression). For the dataset, the **public benchmark datasets** were mainly used for comparison with other studies, and other datasets were also used. **Data split of training/validation/test proceeds in the same way as corresponding comparative studies.** Since **the hyper-parameters of TabNet optimized for each task are all different**, the appendix of the preprint mentioned the details of model set-up.

Instance-wise Feature Selection

Model	*Test AUC*					
	Syn1	Syn2	Syn3	Syn4	Syn5	Syn6
No selection	.578 ± .004	.789 ± .003	.854 ± .004	.558 ± .021	.662 ± .013	.692 ± .015
Tree	.574 ± .101	.872 ± .003	.899 ± .001	.684 ± .017	.741 ± .004	.771 ± .031
Lasso-regularized	.498 ± .006	.555 ± .061	.886 ± .003	.512 ± .031	.691 ± .024	.727 ± .025
L2X	.498 ± .005	.823 ± .029	.862 ± .009	.678 ± .024	.709 ± .008	.827 ± .017
INVASE	**.690 ± .006**	.877 ± .003	**.902 ± .003**	**.787 ± .004**	.784 ± .005	.877 ± .003
Global	.686 ± .005	.873 ± .003	.900 ± .003	.774 ± .006	.784 ± .005	.858 ± .004
TabNet	.682 ± .005	**.892 ± .004**	.897 ± .003	.776 ± .017	**.789 ± .009**	**.878 ± .004**

Fig 6. Area under curve (AUC) results on 6 synthetic datasets from the previous study. The first column on the left lists the feature selection-based comparative models. (Source)

The authors experimented using synthetic datasets presented in existing studies. Some datasets are in such a case that only a subset of the features determine the output, while the others are the cases which have salient features which are instancely dependent. For example, the output of **Syn2** is determined based on features $X_3 - X_6$. On the other hand, the output of **Syn4** depends on either $X_1 - X_2$ or $X_3 - X_6$, which both depend on the value of X_{11}. Note that the tests are the classification task.

The comparison models are as follows.

- **No Selection**: Using all features without any feature selection.
- **Tree**: Tree Ensemble model.

- **Lasso-regularized:** Lasso-regularized model
- **L2X:** A feature selection model from the study which presents the synthetic datasets.
- **INVASE:** A neural network-based feature selection model from the existing study.
- **Global:** Using only globally-salient features. (Human-based selection)

The results in the **Fig 6.** above show that TabNet outperforms other existing methods except INVASE, which is the latest technique at the time, in terms of feature selection. The performance is better for **Syn4-Syn6** with different important features for each instance than for **Syn1-Syn3** with the same salient features regardless of each instance.

Performance on Real-World Datasets

Model	*Test accuracy (%)*
XGBoost	89.34
LightGBM	89.28
CatBoost	85.14
AutoML Tables	94.95
TabNet	**96.99**

(a) Forest Cover Type

Model	*Test accuracy (%)*
DT	50.0
MLP	50.0
Deep neural DT	65.1
XGBoost	71.1
LightGBM	70.0
CatBoost	66.6
TabNet	**99.2**
Rule-based	100.0

(b) Poker Hand

Model	*Test acc. (%)*	*Model size*
Sparse evolutionary MLP	**78.47**	**81K**
Gradient boosted tree-S	74.22	0.12M
Gradient boosted tree-M	75.97	0.69M
MLP	78.44	2.04M
Gradient boosted tree-L	76.98	6.96M
TabNet-S	78.25	81K
TabNet-M	**78.84**	**0.66M**

(c) Higgs Boson

Fig 8. Performance results of 3 real-world cases with several comparison models. All three of these tests are classification tasks. (Source)

There were several experiments with **real-world datasets.** The description of datasets are below.

- **Forest Cover Type:** Dataset that **classifies forest types through tree categories.**
- **Poker Hand**: Dataset **ranked through poker hands**.
- **Sarcos**: Dataset of **regressing inverse dynamics of an anthropomorphic robot arm.**
- **Higgs Boson**: Dataset which **distinguishing between a Higgs bosons process vs. background**.
- **Rossmann Store Sales**: Dataset that **predicts store sales through static and time-varying data**.

The authors adopt existing machine learning-based methodologies as comparative models (e.g. XGBoost, LightGBM, etc.). In the Forest classification test, the authors additionally used **Google's AutoML Tables** and in the poker hand problem case, the **Rule-Based Model** was set as the standard, which always returns correct answers. In addition, for the Higgs Boson dataset, the authors experimented with changing the size of the TabNet (*TabNet-S*, *TabNet-M*).

In **Fig 8.**, all three comparison experiments prove that the proposed **TabNet is superior to the existing machine learning model.** In particular, in the case of the Poker Hand problem, it shows much better accuracy than other models. However, the accuracy in the Higgs Boson task does not improve as much as the size of the TabNet model increases.

Model	*Test MSE*	*Model size*
Random forest	2.39	16.7K
Stochastic DT	2.11	28K
MLP	2.13	0.14M
Adaptive neural tree	1.23	0.60M
Gradient boosted tree	1.44	0.99M
TabNet-S	**1.25**	**6.3K**
TabNet-M	**0.28**	**0.59M**
TabNet-L	**0.14**	**1.75M**

(a) Sarcos

Model	*Test MSE*
MLP	512.62
XGBoost	490.83
LightGBM	504.76
CatBoost	489.75
TabNet	**485.12**

(b) Rossmann Store Sales

Fig 8. Performance results of 2 real-world cases with several comparison models. These two tests are regression tasks. (Source)

The regression task is also tested by comparing it with existing machine learning models. The MSE results are shown in **Fig 9**. For regression problems, TabNet also shows superior performance. Especially, the **accuracy improves significantly as the size of the model increases**, contrary to the classification tasks.

Interpretability

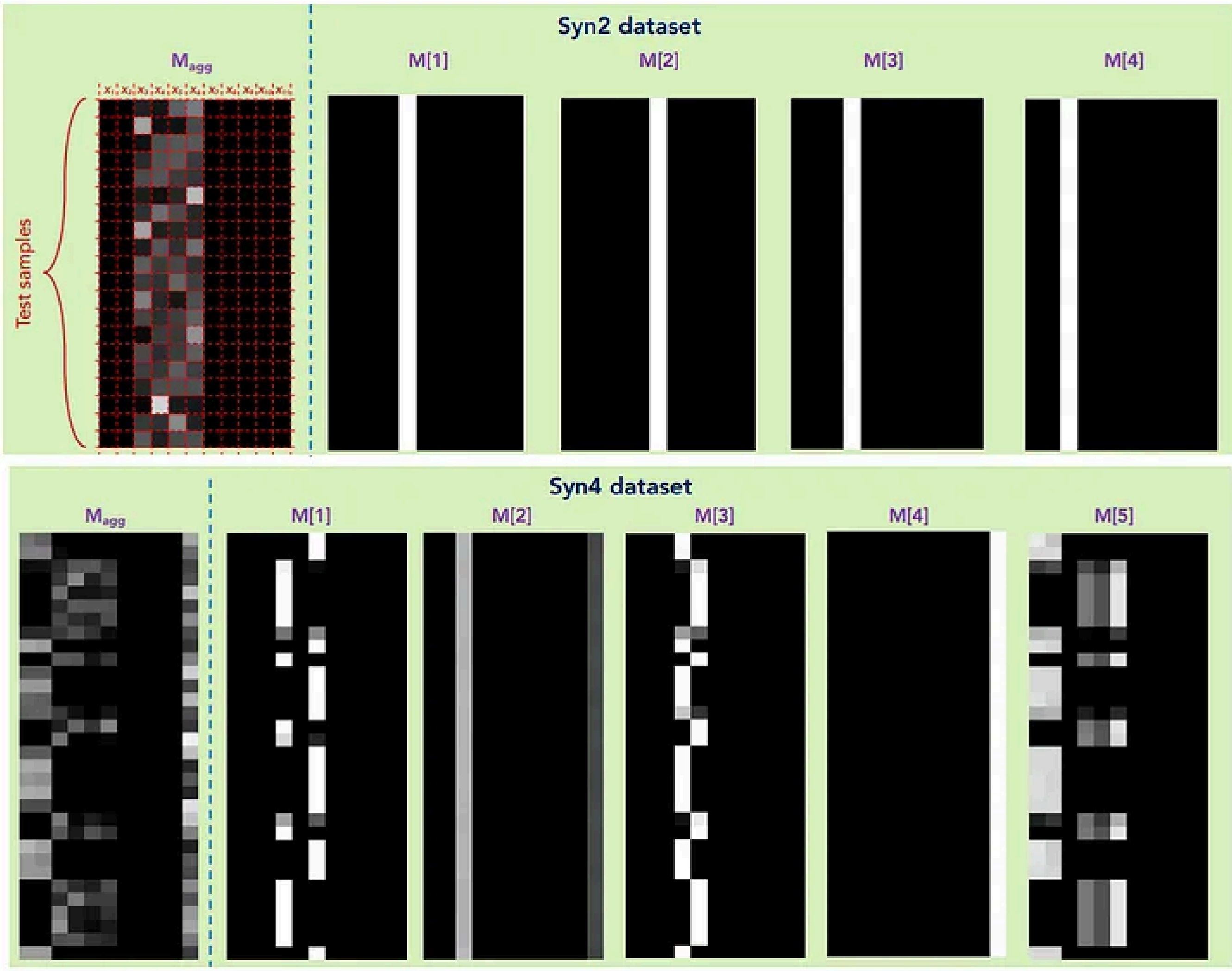


Fig 9. Feature importance masks **M[i]**. These masks are from the experiment with **Syn2** and **Syn4** Dataset. Bright colors indicate high values, meaning they are just as important. (Source)

The authors check how the model extracts features through masks. As mentioned, the mask of TabNet can explain the reason for returned output. The mask results for the two synthesized datasets (**Syn2**, **Syn4**) are shown in **Fig 10.** Syn2 dataset has the same important features regardless of instance, and **Syn4** dataset has different salient characteristics for each instance. The mask descriptions in **Fig 10.** show that the model users can infer feature importance via masks.

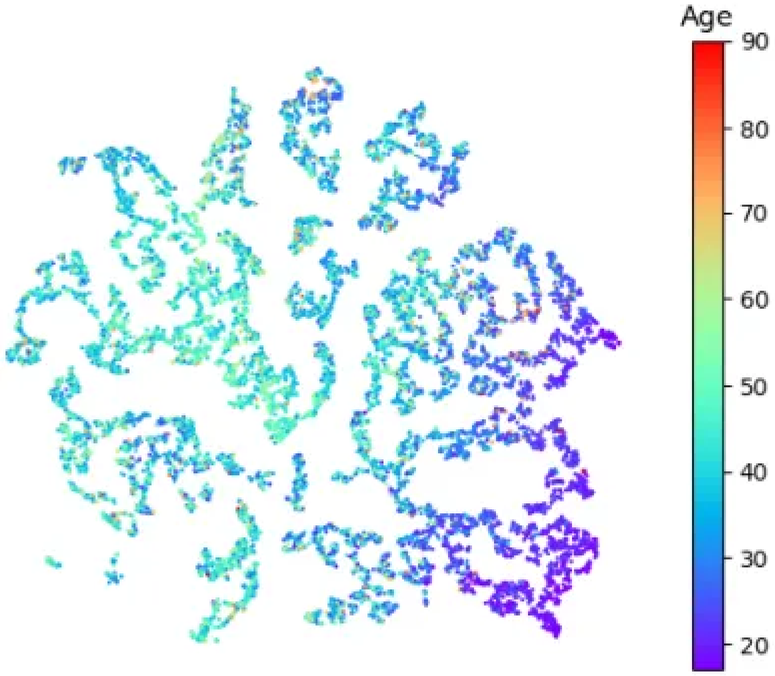


Fig 11. First two dimensions of the T-SNE of the decision manifold created by TabNet from Adult Census Income dataset. This shows the impact of the top feature 'Age'. (Source)

The authors show the experimental results on the actual dataset "Adult Census Income" in **Fig 11**. This figure plots the T-SNE manifold for the "Age" variable, which is considered to have the greatest value on the feature importance evaluated by TabNet. Instance groups are well segregated according to the most important variable "Age".

Self-supervised Learning

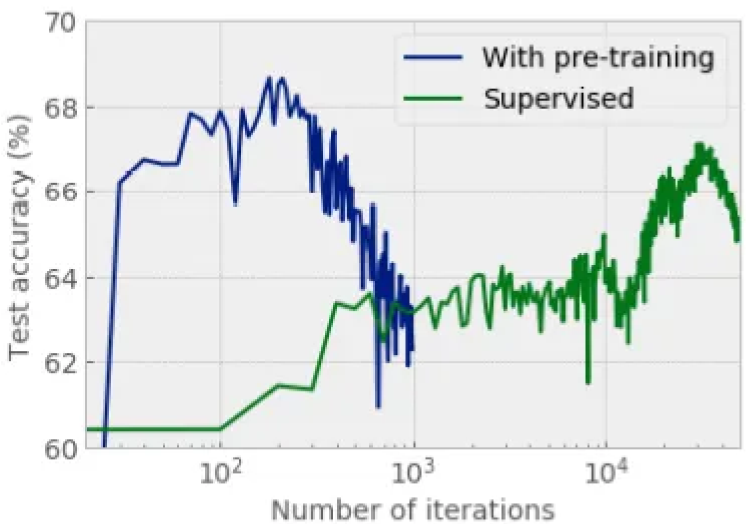


Fig 12. Training curves on Higgs Boson dataset with 10,000 samples. (Source)

Training dataset size	*Test accuracy (%)*	
	Supervised	*With pre-training*
1k	57.47 ± 1.78	$\mathbf{61.37 \pm 0.88}$
10k	66.66 ± 0.88	$\mathbf{68.06 \pm 0.39}$
100k	72.92 ± 0.21	$\mathbf{73.19 \pm 0.15}$

Fig 13. Higgs Boson test results with TabNet-M model. (Source)

The authors evaluate whether TabNet can be utilized for the purpose of self-supervised learning. In the experiment, the model undergoes unsupervised pre-training. Then, comparison experiments are designed between the pre-trained model and the vanila model (trained from the raw-stage). The test accuracies are shown in **Fig 12** and **Fig 13**.

The results in **Fig 12** and **Fig 13** show that the model with the pretext task (pre-training) has better accuracy than the vanila model. In particular, pre-trained model shows better performance with fewer iterations. Moreover, the difference in performance is larger in the situation where the training data is small. **The authors show that the proposed TabNet can also be used as a feature extractor.**

Conclusion

TabNet, a **novel deep learning model for tabular data**, is introduced as an alternative to traditional machine learning approaches. Unlike conventional tabular data models, which primarily rely on **decision trees**, TabNet leverages a **sophisticated neural network architecture**. It selects features in a manner similar to **decision trees** but introduces a **trainable mask mechanism**, allowing the model to **highlight the basis for its predictions**.

Extensive experimental results demonstrate that **TabNet not only delivers strong predictive performance** but also offers **versatility for various tasks**, making it a valuable contribution to deep learning-based tabular data processing.

Review

The authors in this paper represents the innovative neural network architecture for tabular data tasks. One of TabNet's key innovations is its **trainable masks**, which determine **which features contribute most to each prediction**. Also, Unlike tree-based models that require handcrafted feature engineering, TabNet operates as an **end-to-end learning system**, enabling the **automatic extraction of meaningful**

representations directly from raw tabular data. I believe that this deep learning technology **could potentially replace existing manners** like trees and XGboost.

Still, concerns remain clear. First of all, TabNet requires **significantly more computational resources** than traditional models like XGBoost or LightGBM. This may limit its practicality in **low-resource environments.** Furthermore, its performance against **other deep learning methods for tabular data** remains under-explored, leaving room for further research.

Despite its challenges, TabNet represents a **significant step forward** in bringing **deep learning to tabular data processing.** Its **fusion of attention-based learning and decision-tree-inspired selection mechanisms** sets a foundation for further advancements in **interpretable deep learning models.**

Reference

TabNet: Attentive Interpretable Tabular Learning

Clap or any comments on this post are welcome!

Tabular Data · Paper Review · Deep Learning · Neural Networks · Self Supervised Learning

Written by Dong-Keon Kim

55 Followers · 2 Following

I summarize, organize, and review AI papers. I also investigate new or emerging AI technologies.

Responses (2)

Write a response

What are your thoughts?

Michal Wydra
Mar 20

Hi, thanks for your work. I will definitely try TabNet in my research.

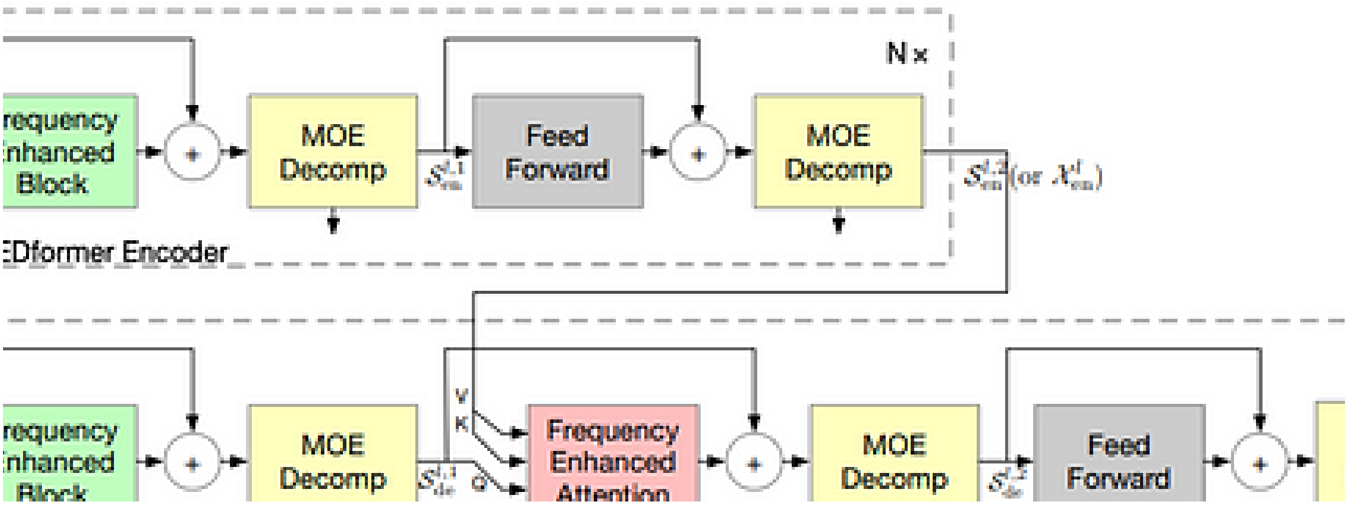
6 1 reply Reply

Valeriy Manokhin, PhD, MBA, CQF
Mar 23

what a joke this model has been around since 2021 and is not SOTA

5 1 reply Reply

More from Dong-Keon Kim

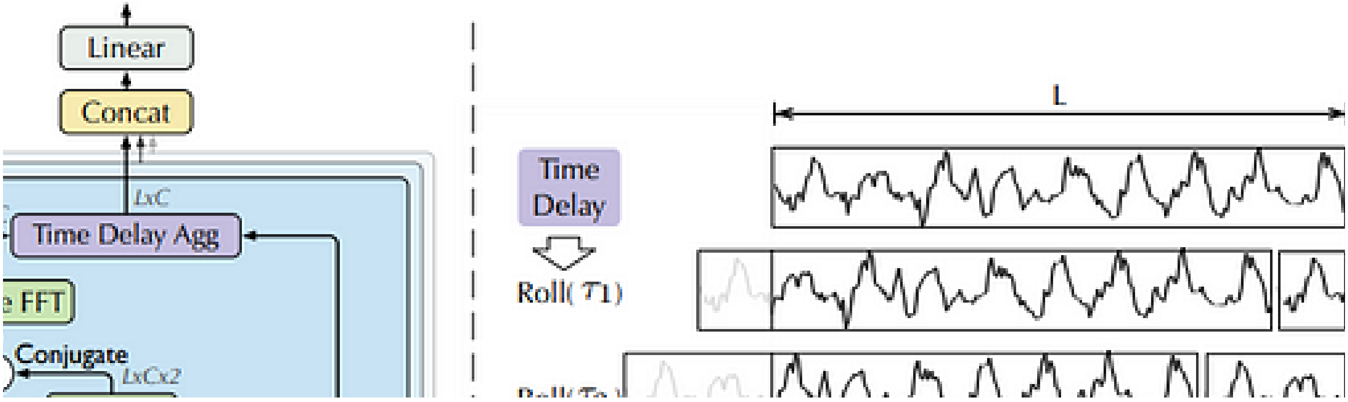


Dong-Keon Kim

FEDformer: Unleashing the Power of Frequency in Time Series Forecasting

A Deep Dive into Frequency Enhanced Decomposed Transformers for Long-Term Time Series Forecasting

Apr 14 22

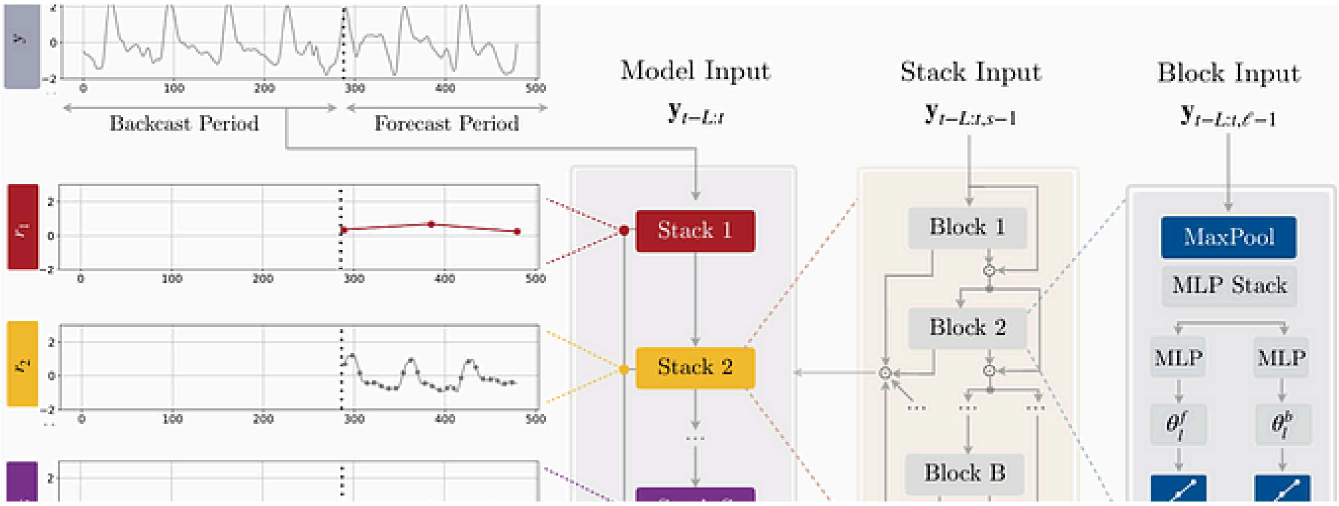


Dong-Keon Kim

Autoformer: Decomposing the Future of Time Series Forecasting

A deep dive into the model that rethinks Transformers with trend-seasonality awareness.

Mar 24 32

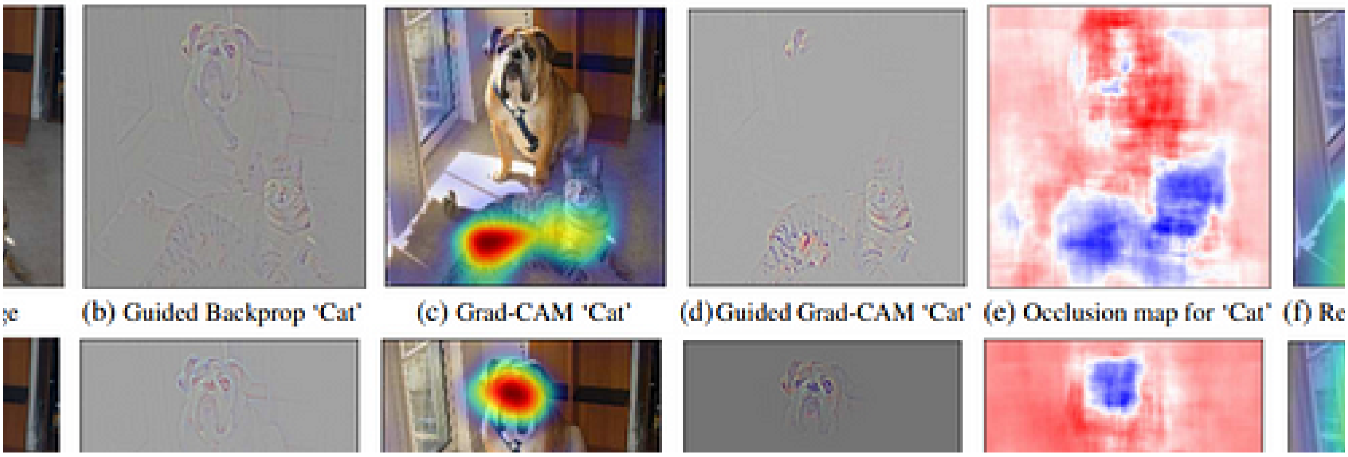


Dong-Keon Kim

NHITS: A New Era of Long-Horizon Time Series Forecasting with Hierarchical Interpolation

A deep dive into how NHITS achieves superior forecasting accuracy and computational efficiency.

4d ago 7



Dong-Keon Kim

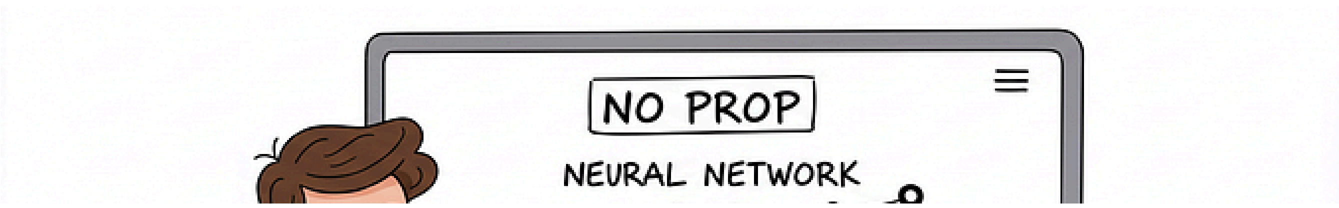
Grad-CAM: A Gradient-based Approach to Explainability in Deep Learning

A Comprehensive Summary and Review of the Grad-CAM Paper

Feb 17 11

See all from Dong-Keon Kim

Recommended from Medium

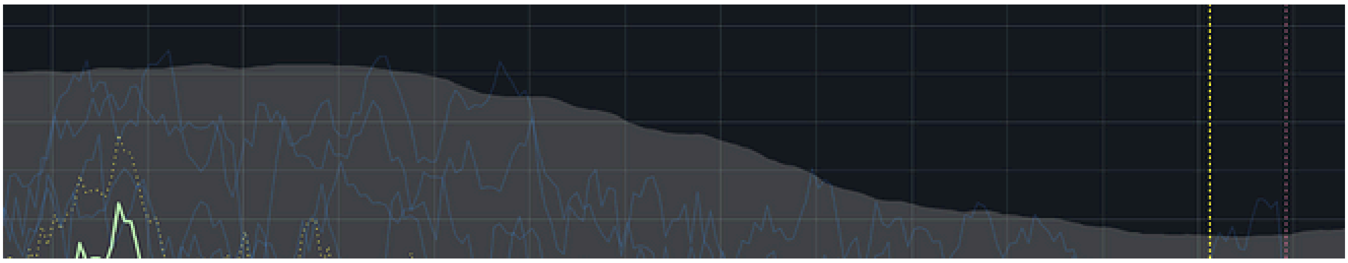


In AI Advances by Dr. Ashish Bamania

You Don't Need Backpropagation To Train Neural Networks Anymore

A deep dive into the 'NoProp' algorithm that eliminates the need for Forward pass and Backpropagation and coding it up from scratch.

6d ago 1K 10



In Booking.com Engineering by Ivan Shubin

Anomaly Detection in Time Series Using Statistical Analysis

Setting up alerts for metrics isn’t always straightforward. In some cases, a simple threshold works just fine—for example, monitoring…

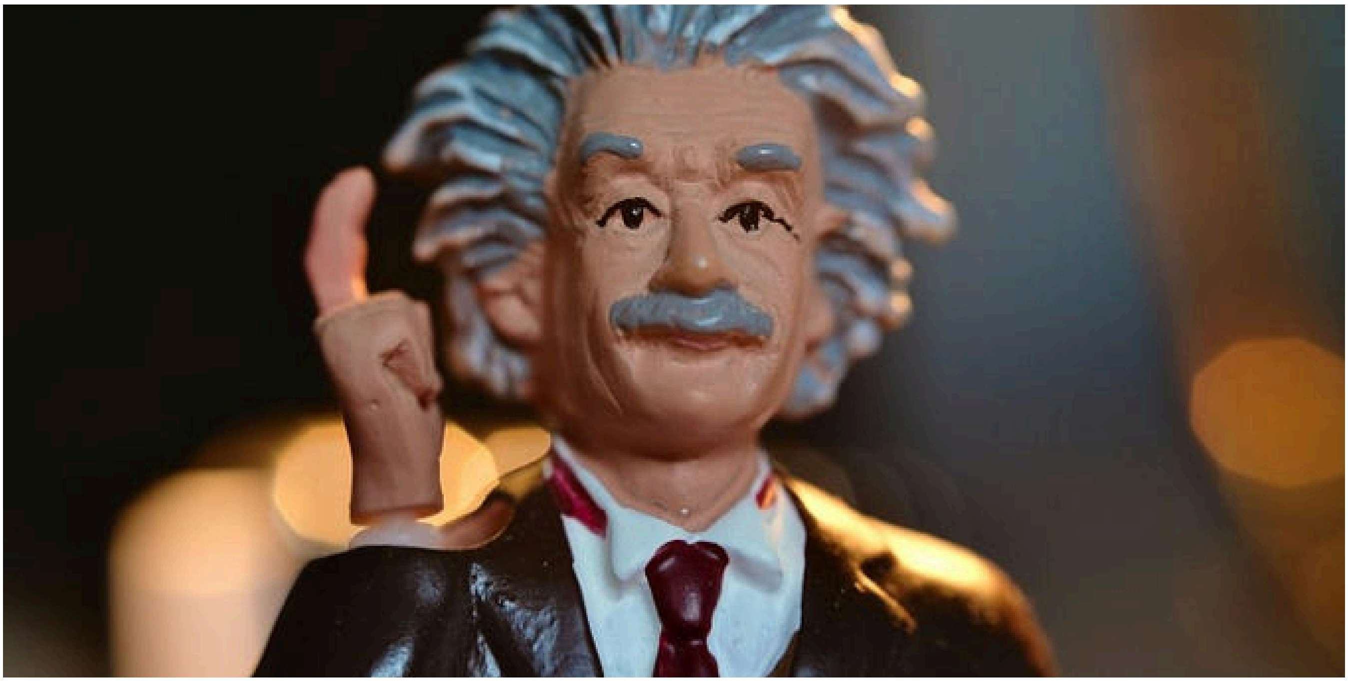
Apr 15 273 5

Shuai Guo, PhD

Physics-Informed Neural Networks for Anomaly Detection: A Practitioner’s Guide

The why, what, how, and when to apply physics-guided anomaly detection

Apr 24 549 11

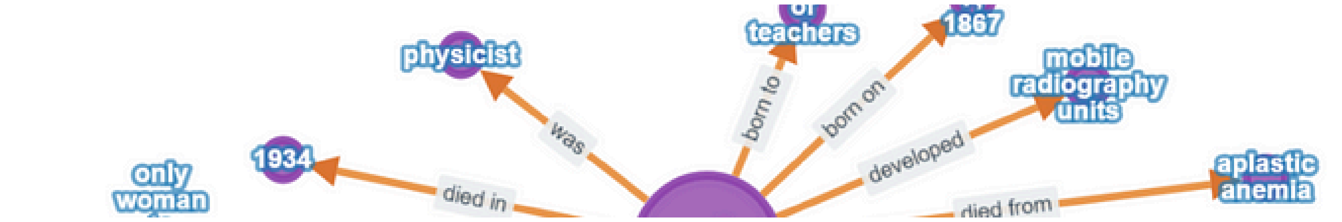


In Data Science in Your Pocket by Mehul Gupta

Google's Data Science Agent: Data Scientists are doomed

Just upload your dataset and build automatic data science pipelines

Mar 4 665 32

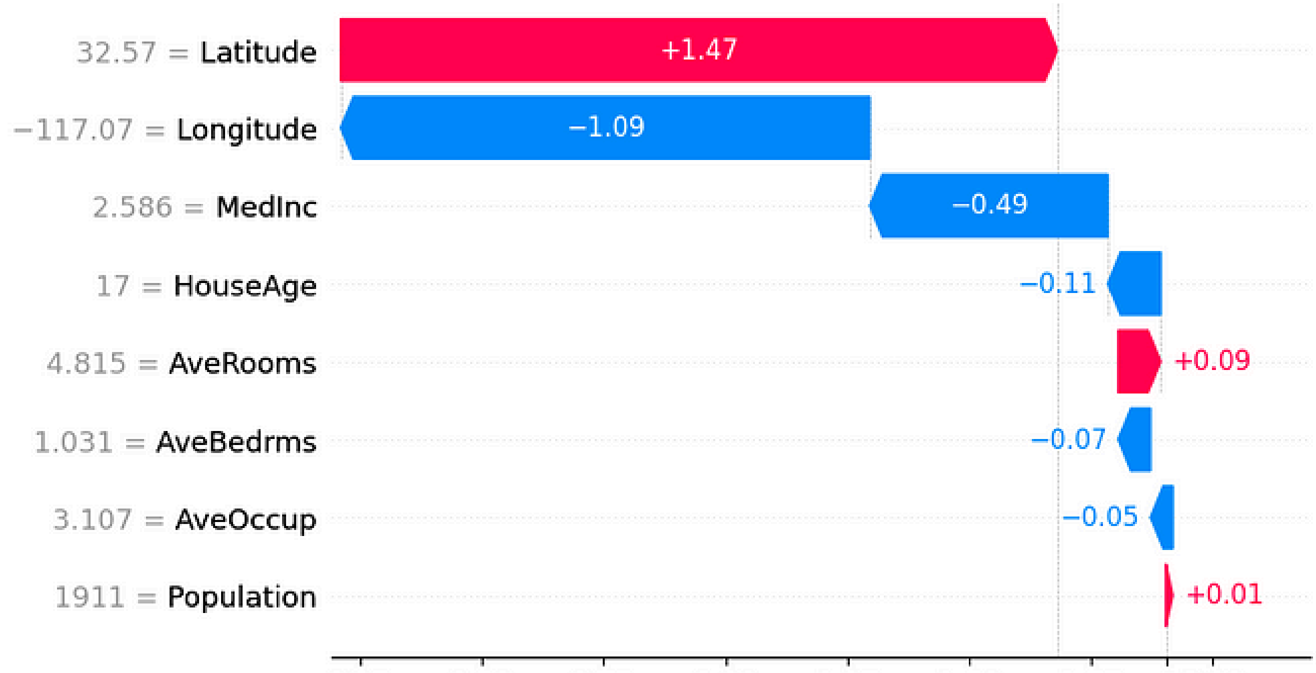


In Level Up Coding by Fareed Khan

Converting Unstructured Data into a Knowledge Graph Using an End-to-End Pipeline

Step by Step guide

Apr 17 · 1.6K · 30



In Towards Dev by Nivedita Bhadra

How to Make Machine Learning Models Explainable Using SHAP

Explain your machine learning model with Shapley values in Python

Oct 31, 2024 · 33

See more recommendations