Tabular Learning with Deep Neural Nets

Jacob Yoke Hong Si [*]
Department of Computer Science
University of Toronto
Toronto, ON M5S 1A1
jacobyh.si@mail.utoronto.ca

Evence YiFan Wang [*]
Department of Computer Science
University of Toronto
Toronto, ON M5S 1A1
evence.wang@mail.utoronto.ca

Sean Seoyoon Lee [*]
Department of Computer Science
University of Toronto
Toronto, ON M5S 1A1
seoyoon.lee@mail.utoronto.ca

Abstract

Deep neural nets were seldom used to learn tabular data in the past. Recently, there are findings that illustrate DNN models with high-performance in this task. Hence, this motivates the paper to evaluate the performance of two novel tabular data learning techniques that incorporate deep neural net architectures, **Tabnet**, a DNN that incorporates sequential attention, and **SAINT**, a transformer that uses contrastive learning to learn tabular data. We retrain these models on tabular datasets to reproduce the results and conduct sensitivity analysis. Lastly, we extend the models to new datasets and determined that Tabnet generalizes better and trains quicker compared to Saint. However, we expect Saint to outperform Tabnet in accuracy with pre-training that we are unable to evaluate due to the lack of computational power.

1 Introduction

Tabular data is one of the most common data forms. A wide array of industries like finance, healthcare, and logistics utilize tabular formatting for their data. DNN (Deep Neural Net) structures have achieved high performance with image, audio, and text (language processing) but have only recently been applied to tabular data. There are many aspects of tabular data which have hindered applications in DNNs previously. One such characteristic is the inclusion of heterogeneous features from a multitude of discrete and continuous data distributions. The correlation or independence of these features could prevent the model from predicting accurately. Additionally, the lack of inherent positional information in the tabular data makes it difficult for neural networks to learn effectively. Recent exploratory research in this area yielded high performing neural net architectures that outperform existing methods in predicting and learning tabular data. In this paper, we compare the performance of two such models: **Tabnet**, a deep neural net which utilizes sequential attention, and **SAINT**, a Self-Attention and Intersample Attention Transformer, by testing their performance on tabular datasets.

[*]All authors contributed equally to the paper

2 Related Works

2.1 Classic Method of Tabular Learning

The objective of tabular learning is to classify or estimate a feature of a row given sufficient information of other features on the same row. For example, we could tell the model to predict the income level of an individual given their occupation, education etc. To achieve high performance, popular non-DL methods of tabular learning like XGBoost employ highly effective feature selection tools to create decision trees to produce accurate predictions (Chen 2016). Decision trees are prevalent in tabular learning due to their synergy with efficient feature selection tools that yield features with the most statistical information gain (Grabczewski 2005). Features with high information gain correspond to features that are more effective in learning correlations in data. For example, the education level of an individual might have high correlation with the income of the individual. Knowing which features to learn enhances the accuracy of the model greatly. XGBoost also incorporates ensembles in its architecture to reduce variance and improve the model. Sometimes, these models are preferred to deep learning alternatives due to their interpretability.

2.2 Attention Architecture

Attention-based architecture is an effective performance enhancer in RNN models. Conceptually, this architecture directs the machine's focus to more relevant and important facets of learning. Attention was originally introduced to address the inability of sequence predictors to accurately process long input sequences. Instead of predicting the next component of a sequence based on the last hidden state in the RNN, attention components retain all the hidden states and produce a context vector which affects the prediction of the next element in the sequence. For example, attention architecture in NLP learn to map key-words and assign heavier weights to those words when processing the input sentence. Similarly, attention components in time series forecasters can learn trends and patterns in data, assigning heavier weights to prominent patterns in the input before forecasting.

3 Model Architecture

3.1 Tabnet

The architecture of Tabnet requires the input of raw tabular data which is encoded using sequential multi-step processing and trained using gradient descent based optimisation. At each decision step, sequential attention is utilised to determine the most salient features used at each decision step to help predict the classification/regression task.

3.1.1 Feature selection

Tabnet employs a learnable feature mask for soft sparse selection of salient features. This enables improved learning since the most salient features would be focused by the model hence, the learning capacity of a decision step is not wasted on less important features, leading to a more parameter efficient model.

In order to apply sparsity to the model, a sparsity regularizer is used which is as follows.

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

where M is the feature mask, N_{steps} is the number of decision steps, B is the batch size, D is the feature dimension and ϵ is a small number for numerical stability. Sparsity improves the model by providing favorable inductive bias for datasets where most features are redundant.

3.1.2 Feature processing

The features are processed using a feature transformer where layers are shared across decision steps. Each fully connected layer is connected to a batch normalization and gated linear unit nonlinearity, followed by a normalized residual connection. In the layers apart from the input features, ghost BN is used where it is comprised of a virtual batch size and momentum. Ghost BN is avoided in the input features since we want to take advantage of low-variance averaging. Lastly, a decision embedding is constructed where ReLU activation is applied and a linear mapping is added in order to obtain the output mapping. The model architecture can be found in Figure 1 of the Appendix.

3.2 SAINT: Self-Attention and Intersample Attention Transformer

The SAINT model is a transformer that utilizes two different types of attention to learn tabular data. The model vectorizes the input data by embedding it into a d-dimensional space and applies a self-attention block coupled with the novel intersample attention block to learn the data. The model is also equipped to perform self-supervised learning through contrastive methods.

Suppose $\mathcal{D} = \{\mathbf{x}_i, y_i\}_{i=1}^{m}$ is the tabular dataset with m rows; where $\mathbf{x}_i$ is an n-dimensional feature vector, and y_i is a label or a target variable. The model also appends a learned embedding token [cls] to each $\mathbf{x_i}$. If $\mathbf{E}$ is the embedding layer that embeds each feature vector to a d-dimensional space, then $\mathbf{E}(\mathbf{x}_i) \in \mathbb{R}^{(n+1)\times d}$.

The SAINT model is comprised of $\mathcal{L}$ stacks of self-attention and intersample attention blocks. A graphic overview of the SAINT neural network architecture is shown in Figure 2 of the Appendix. The encoder has a multi-head (h heads) self-attention layer (MSA), two fully-connected feed-forward layers (FF) with a GELU non-linearity. Each layer has a skip connection and layer normalization (LN). The multi-head intersample attention block (MISA) follows a similar structure to the multi-head self attention block above; however, the self attention layer is replaced by an intersample attention layer. See Appendix A for an algebraic derivation of the SAINT output.

Intersample attention is a type of row attention where it is computed across different data points or rows of a tabular data matrix in a given batch rather than just the features of a single data point. This model improves the representation of a given row by inspecting the other rows. It also handles null values well by borrowing corresponding features from similar data samples in the batch. (See Appendix B for the pseudocode)

SAINT also uses contrastive learning to conduct self-supervision. Standard contrastive learning methods in vision form different "views" of images using crops and flips (Chen T., et al. 2020). The SAINT model uses Cutmix to augment samples in the input space and mixup for the embedding space. The combination of these methods create an effective self-supervision task for the SAINT network. (See Appendix C for the detailed procedure)

The model uses cross entropy loss or mean squared error loss to denoise the contrastive learning outputs. It also uses InfoNCE loss to compare the projected heads (z_i, z_i') of the SAINT block outputs. The cumulative loss function for this procedure is shown below:

$$\mathcal{L}_{pretraining} = -\sum_{i=1}^{m} \log \frac{exp(z_i z_i'/\tau)}{\sum_{k=1}^{m} exp(z_i z_k'/\tau)} + \lambda_{pt} \sum_{i=1}^{m}\sum_{j=1}^{n} \left[\mathcal{L}_j(MLP_j(\mathbf{r}'_i), \mathbf{x_i})\right]$$

where $\mathbf{r}'_i$ is the contrastive learning version of the output from the SAINT layers, z_i is the projection head applied to the output $\mathbf{r}_i$ and z_i' is the projection head applied to the output $\mathbf{r}'_i$, $\mathcal{L}_j$ is cross-entropy loss or mean squared error depending on the j^{th} feature being categorical or continuous, MLP_j is a single hidden-layer perceptron with a ReLU non-linearity and lastly, λ_{pt} and τ (temperature) are hyperparameters that can be tuned using validation data. Finally, after the pre-training, the [cls] token is passed into the single hidden-layer MLP with ReLU activation to obtain the final output $\hat{y}_i$.

4 Experiments and Discussions

4.1 Sensitivity Analysis Overview

We conduct sensitivity analyses for both models by fixing a complement set of hyperparameters to their default values and changing the hyperparameter we are interested in. With each model, we report the best settings we have discovered. The optimal hyperparameters and results can be found in Appendix D and E respectively.

4.1.1 Tabnet

Conducting sensitivity analysis on Tabnet, we evaluate the accuracy of the model when predicting the cover type of the forest dataset. We select several key variables to perform sensitivity analysis on. This includes N_d (feature dimension), N_a (output dimension), learning rate, batch size and λ_{sparse}. We set $N_d = N_a$ since the paper indicated it to be a "reasonable choice for most datasets". As N_d and N_a increases, accuracy of the model improves. However, a very high value of N_d and N_a could lead to overfitting. In terms of learning rate, we obtain the best accuracy at learning rates between 0.02 and 0.025. Thus, a learning rate should not be too low as training could get stuck in an undesirable local minimum whereas a learning rate that is too high could lead to divergence whilst performing gradient optimization. With regards to λ_{sparse}, we observe that a λ_{sparse} of 0.0001 generalizes best. Lastly, a smaller batch size of 256 yields the highest accuracy for the model. This aligns with the trend where batch sizes that are too large will lead to poorer generalizations and that smaller batch sizes appear to converge quicker onto more accurate solutions ("Effect of batch size on training dynamics", 2022). Utilizing the aforementioned optimal hyperparameters, we have managed to reproduce the results with a 94.4% accuracy which is close to the paper's 96.99%.

4.1.2 SAINT

To conduct sensitivity analysis for SAINT, we gather hyperparameters outlined in the paper such as learning rate, batch size, and embedding size. We also test for temperature and the dropout rates that the SAINT model specifies. We use the default settings outlined in the SAINT paper and only modify the hyperparameter of interest to conduct sensitivity analysis.

The learning rate of 0.0001 achieves the highest validation accuracy compared to larger learning rates. The best batch size we found for SAINT obtained is 128. A batch size of 512 yields similar results, but is far less cost efficient. The highest performing embedding size is 16. Higher embedding sizes may perform better, but 16 is the maximum size with our limited computing power. The best temperature for SAINT is 0.5 or 0.3 which conflicts with the training settings outlined by the paper ($\tau = 0.7$). With the settings above, we have reproduced the results on the Arrhythmia dataset with 90.2% AUROC compared to the reported AUROC value in the original paper: 94.18%. We were unable to pre-train the model as described in the original paper due to the lack of compute; however, with better computational resources, we are confident that we would be able to reproduce higher AUROC scores.

4.2 Extension to New Dataset & Performance Analysis

To extend Tabnet and Saint, we train them on the adult dataset and evaluate their performance. The adult dataset is a multivariate tabular dataset that is fit for classification tasks. The dataset also features both continuous and categorical datatypes like capital-gain and education level. We choose this dataset because it embodies the characteristics of tabular data. The performance of the two models are very competitive. Tabnet reported 88.09% accuracy and SAINT reported 86.40% accuracy. The original paper suggests that SAINT outperforms Tabnet. Although SAINT would be more accurate, Tabnet is quicker to train. Tabnet only requires 3 to 5 seconds per epoch whereas SAINT needs almost a minute for each epoch excluding pre-training. Moreover, SAINT suffers from high memory requirements. Tabnet is quite competitive against SAINT; what it lacks in accurate performance is made up for with its speed and low computational cost. The reason SAINT has high computational cost is most likely the vector embedding method SAINT utilizes in its architecture. Higher embedding dimensions make the model more accurate at a heavy cost.

5 Conclusion

In this paper, we compare the structure and performance of two novel models for tabular learning: Tabnet and SAINT. Tabnet uses sequential attention and feature-based transformers, while SAINT uses a combination of self-attention and intersample attention to learn tabular data. Both models show competitive performance on the adult dataset. We observe that Tabnet generalizes better compared to SAINT without pre-training. However, we expect that SAINT with pre-training will perform better, but at a heavy computational price.

References

[1] Chen, T., Guestrin, C. (2016) Xgboost: A scalable tree boosting system. In Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining, pages 785–794 [Accessed 16 April 2022].

[2] Chen, T., Kornblith, S., Norouzi, M., Hinton, G. (2020) A simple framework for contrastive learning of visual representations. In International conference on machine learning, pages 1597–1607. PMLR. [Accessed 16 April 2022]

[3] Grabczewski, K., Jankowski, N. (2005) Feature selection with decision tree criterion. In HIS. [Accessed 16 April 2022].

[4] Somepalli, G., Goldblum, M., Schwarzschild, A., Bruss, C. B., Goldstein, T. (2021) SAINT: Improved neural networks for tabular data via row attention and contrastive pre-training. arXiv preprint arXiv:2106.01342. [Accessed 16 April 2022]

[5] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A., Kaiser, L., Polosukhin, I. (2017) Attention is all you need. arXiv preprint arXiv:1706.03762. [Accessed 16 April 2022]

[6] Arık, S. O., Pfister, T. (2021) Tabnet: Attentive interpretable tabular learning. In AAAI (Vol. 35, pp. 6679-6687).

[7] Effect of batch size on training dynamics. Medium. (2022). Retrieved 18 April 2022, from https://medium.com/mini-distill/effect-of-batch-size-on-training-dynamics-21c14f7a716e.

Figures

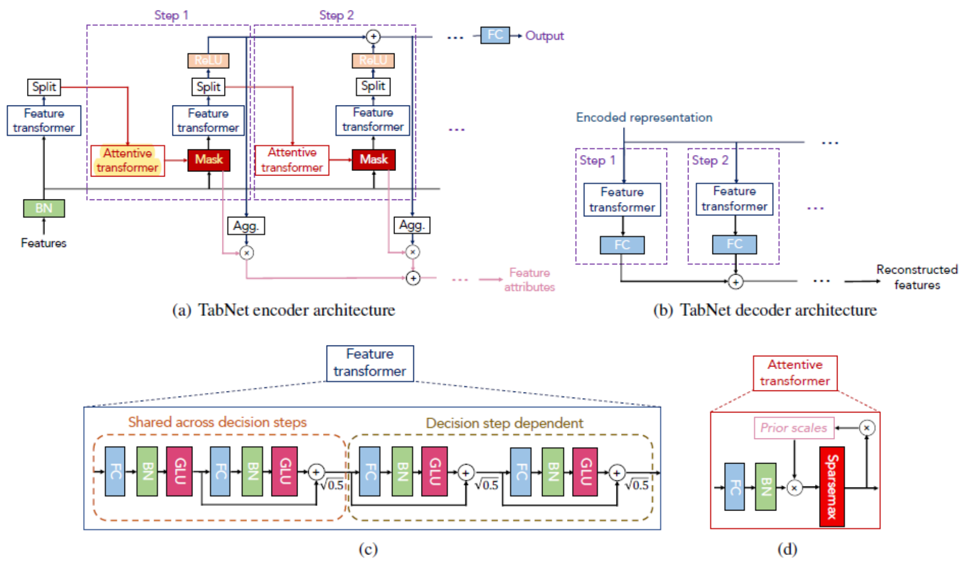

Figure 1: Tabnet architecture outline (Arik et al. 2021)

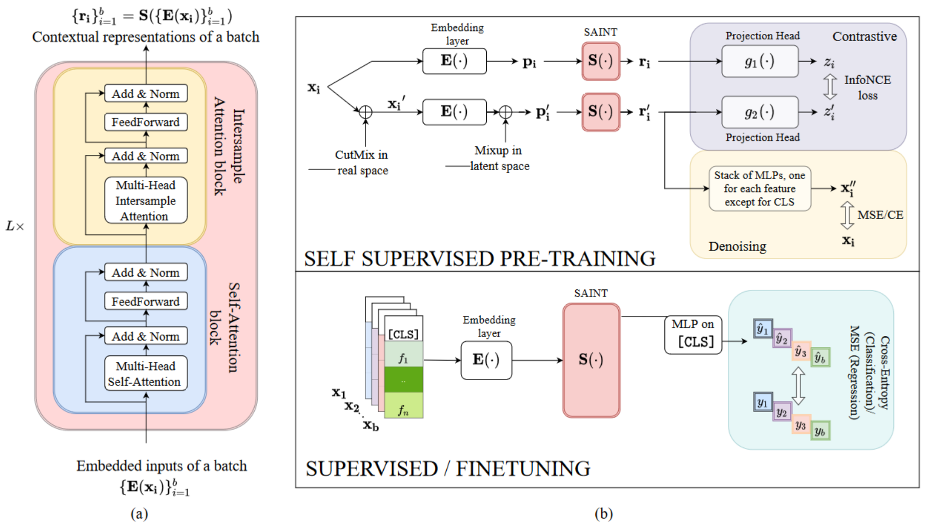
Contextual representations of a batch
Add & Norm
FeedForward
Multi-Head Intersample Attention
Intersample Attention block
Multi-Head Self-Attention
Self-Attention block
L×
Embedded inputs of a batch
(a)
Embedding layer
SAINT
Projection Head
Contrastive
InfoNCE loss
CutMix in real space
Mixup in latent space
Stack of MLPs, one for each feature except for CLS
MSE/CE
Denoising
SELF SUPERVISED PRE-TRAINING
MLP on [CLS]
Cross-Entropy (Classification)/ MSE (Regression)
SUPERVISED / FINETUNING
(b)

Figure 2: SAINT architecture outline (Somepalli et al. 2021)

Appendix

A: Neural Net Structure of the Saint Block

$\mathbf{r}_i$ or SAINT output derivation:

$$\begin{aligned}
\mathbf{z}_i^{(1)} &= \mathbf{LN}(\mathbf{MSA}(\mathbf{E}(\mathbf{x}_i))) + \mathbf{E}(\mathbf{x}_i) \\
\mathbf{z}_i^{(2)} &= \mathbf{LN}(\mathbf{FF}_1(\mathbf{z}_i^{(1)})) + \mathbf{z}_i^{(1)} \\
\mathbf{z}_i^{(3)} &= \mathbf{LN}(\mathbf{MISA}(\{\mathbf{z}_i^{(2)}\}_{i=1}^{b})) + \mathbf{z}_i^{(2)} \qquad \text{(b is batchsize)} \\
\mathbf{r}_i &= \mathbf{LN}(\mathbf{FF}_2(\mathbf{z}_i^{(3)})) + \mathbf{z}_i^{(3)}
\end{aligned}$$

B: Pseudocode for both self-attention and intersample attention

```
# b : batch size, n : number of features, d : embedding dimension
# W_q, W_k, W_v are weight matrices of dimension dxd
# mm : matrix - matrix multiplication
def self_attention(x):
    # x is bxnxd
    q, k, v = mm(W_q, x), mm(W_k, x), mm(W_v, x) #q ,k , v are bxnxd
    attn = softmax(mm(q, np.transpose(k, (0, 2, 1))) / sqrt(d)) # bxnxn
    out = mm(attn, v) # out is bxnxd
    return out

def intersample_attention(x):
    # x is bxnxd
    b, n, d = x.shape # as mentioned above
    x = reshape (x, (1, b, n * d)) # reshape x to 1xbx(n * d)
    x = se lf_att ention ( x ) # the output x is 1xbx(n * d)
    out = reshape (x, (b, n, d)) # out is bxnxd
    return out
```

C: Contrastive Learning Algebra

Assume that l out of m datapoints are labeled.
$\mathbf{x'}_i$ is the CutMix in the raw dataspace.
$\mathbf{p'}_i$ is the mixup in the embedding space.
$\mathbf{m}$ is the binary mask vector sampled from a bernoulli distribution with $p = p_{CutMix}$.

$$\mathbf{x'}_i = \mathbf{x}_i \odot \mathbf{m} + \mathbf{x}_a \odot (\mathbf{1} - \mathbf{m}) \qquad (\mathbf{x}_a \text{ and } \mathbf{x}_b \text{ is a random sample from the batch})$$
$$\mathbf{p'}_i = \alpha * \mathbb{E}(\mathbf{x'}_i) + (1-\alpha) * \mathbb{E}(\mathbf{x'}_b) \qquad (\mathbf{x'}_b \text{ is a cutmix version of } \mathbf{x}_b)$$

D: Optimal Hyperparameters Used

Tabnet Forest Dataset (reproduction):

$N_D = 128$, $N_A = 128$, $N_{steps} = 3$, $\lambda_{sparse} = 0.0001$, learning rate = 0.025, batch size = 256, epochs = 50

Tabnet Adult Dataset (dataset extension):

$N_D = 32$, $N_A = 32$, $N_{steps} = 4$, $\lambda_{sparse} = 0$, learning rate = 0.02, batch size = 256, epochs = 50

Saint Arrhythmia Dataset (reproduction):

lr = 0.0001, bs = 128, attention dropout = 0.8, embedding size = 16, temperature = 0.5

Saint Adult Dataset (dataset extension):

lr = 0.0001, bs = 128, attention dropout = 0.1, embedding size = 16, temperature = 0.3

E: Sensitivity Analysis

Tabnet Forest Dataset

We use 5 epochs when conducting the sensitivity analysis due to limited computing power.

λ_{sparse}	0	0.0001	0.001	0.01	0.1
Validation Acc.	86.84	**88.05**	86.66	87.72	85.19

N_d and N_a	16	24	32	64	128
Validation Acc.	80.15	80.93	80.02	82.28	**87.05**

Learning Rate	0.005	0.01	0.02	0.025	0.05
Validation Acc.	81.96	84.15	87.05	**87.50**	86.71

Batch Size	256	512	1024	2048	4096
Validation Acc.	**88.05**	84.29	87.50	83.50	74.99

SAINT Arrhythmia Dataset

We use 10 epochs when conducting the sensitivity analysis due to limited computing power.

Learning Rate	0.00001	0.0001	0.001	0.01	0.1
Validation Acc.	73.68	**76.24**	56.44	43.86	56.14

Batch Size	32	64	128	256	512
Validation Acc.	82.47	80.70	**85.97**	81.19	85.63

Embedding Dim.	2	4	8	16
Validation Acc.	78.95	84.21	85.97	**87.61**

Dropout Rate	0.8	0.1
Validation Acc.	**84.21**	78.95

Temperature	0.3	0.4	0.5	0.6	0.7
Validation Acc.	**84.21**	82.46	**84.21**	82.46	83.15

F: Contributions

- Jacob Yoke Hong Si: Reproduced and Extended Tabnet, Wrote Tabnet sections and reviewed the paper
- Evence YiFan Wang: Reproduced and Extended Saint as well as reviewed the paper
- Sean Seoyoon Lee: Wrote overall body of the paper as well as reviewed the paper