Predicting Blood Transfusions for Coronary Artery Bypass Graft Patients using Deep Neural Networks and Synthetic Data

Hsiao-Tien Tsai[1] · Jichong Wu[1] · Puneet Gupta[2] · Eric R Heinz[2] · Amir Jafari[1]

Abstract

Coronary Artery Bypass Graft (CABG) is a common cardiac surgery but continues to have many associated risks, including the need for blood transfusions. Previous research has shown that blood transfusion during CABG surgery is associated with an increased risk for infection and mortality. The current study aims to use modern techniques, such as deep neural networks and data synthesis, to develop models that can best predict the need for blood transfusion among CABG patients. Results show that neural networks with synthetic data generated by DataSynthesizer has the best performance. Implications of results and future directions were discussed.

Keywords deep neural network · data synthesis · blood transfusion · cardiac surgery

✉ Hsiao-Tien Tsai
jennytsai@gwu.edu

Jichong Wu
jwu36@gwu.edu

Puneet Gupta MD
gupta14@gwu.edu

Eric R Heinz MD PhD
eheinz@mfa.gwu.edu

Amir Jafari PhD
ajafari@gwu.edu

[1] Data Science, George Washington University, DC, United States

[2] Department of Anesthesiology and Critical Care Medicine, George Washington University School of Medicine and Health Sciences, DC, United States

1 Introduction

Coronary Artery Bypass Graft (CABG) is a common cardiac surgery but continues to have many associated risks, including infection, major bleeding, and the need for blood transfusion (Horvath, Acker, Chang, & Bagiella et al, 2013) (Kaur, et al., 2021) (Li, et al., 2024). Blood transfusion is a life-saving intervention in patient care, however, previous research has shown that blood transfusion during CABG surgery is associated with an increased risk for infection and mortality after surgery (Horvath, Acker, Chang, & Bagiella et al, 2013). Specially, post-operative blood transfusion after CABG is associated with higher odds of readmission and heart failure within 30-days (Mufarrih, Mahmood, Qureshi, Yunus, & Matyal et al, 2023).

To lower the risk of mortality after surgery, there is a need to develop models that preoperatively predict which patients will need an intra-operative or post-operative blood transfusion. This will not only help to improve patient selection and patient education, but also physician preoperative awareness and perioperative guidelines for CABG patients. Therefore, the goal of this research is to explore modern data analysis techniques and find the models that can best make predictions, such as deep neural networks and synthetic data generation.

2 Related Work

2.1 Cardiac Surgery and Blood Transfusion

Research have been conducted to investigate factors that can help to predict major bleeding (Gao, et al., 2022) and the need for red blood cell transfusion after cardiac surgery (Li, et al., 2024). In one of the most relevant studies (Tschoellitsch, Bock, Mahecic, Hofmann, & Meier, 2022), the researchers employed machine learning models to predict perioperative allogeneic blood transfusion for cardiac patients, where the best model (Random Forest) showed good performance (AUC ranged from .76 - .86); however, the study has several limitations. For example, the data was from a single adult cardiac surgery center in Austria with a relatively small sample size (N = 3,782), thus the results may not be generalizable to other samples with more diverse demographic backgrounds or nationalities. Moreover, the studies only predicted allogeneic blood transfusion (i.e., transfusion of more than 10 units of packed red blood cells; pRBC), while blood transfusion has been reported to be associated with many known risks regardless of volume. Lastly, the study only tested the basic machine learning models (e.g., tree-based models), and it is likely that the performance can be significantly improved using more advanced techniques and sophisticated models. For example, synthetic data generation techniques have been widely used to train and test neural networks, especially when the data has privacy concerns such in domains such as healthcare and employment (Kaur, et al., 2021).

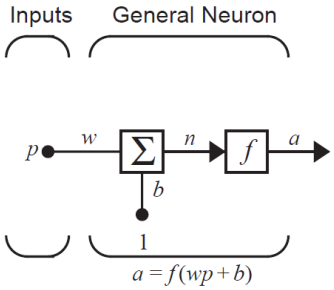
To address this research gap, the current research used the national medical database in the U.S. with a large sample size of over 13,500 data points. Additionally, we predicted the need for blood transfusion regardless of volume. Lastly, we experimented with various modern data analytic approaches to optimize the performance, including deep neural networks and synthetic data generation.

2.2 Artificial Neural Networks

2.2.1 The Basics of Neural Networks

An artificial neural network (ANN) is a machine learning program inspired by the structure and functions of human brains. It trains the computer to process data and perform tasks using interconnected nodes, or neurons, in a layered structure. This section first introduces basic concepts and most popular neural network architectures (Martin T. Hagan, 2014) based on a review of literature, followed by a summary different neural network architecture to process tabular data using neural network deep learning. A neural network is typically consisted of neurons which are connected with weights (coefficients), which together can be called as a layer. These layers of neurons are connected in a network and this is where the power of neural computation comes from. Each layer hosts the weighted inputs, transfer function and produce an output. The performance of a neural network is determined by the transfer functions of its neurons, by the learning rule and by the architecture itself. The weighed sum of the inputs constitutes the activation of the neuron. A neural network is a parameterized system with three major adjustable parameters: the interconnection pattern between layers of neurons, the learning process for updating the weights, and the activation function that converts a neuron’s weighted input to output.

Figure 1 Single-input neuron



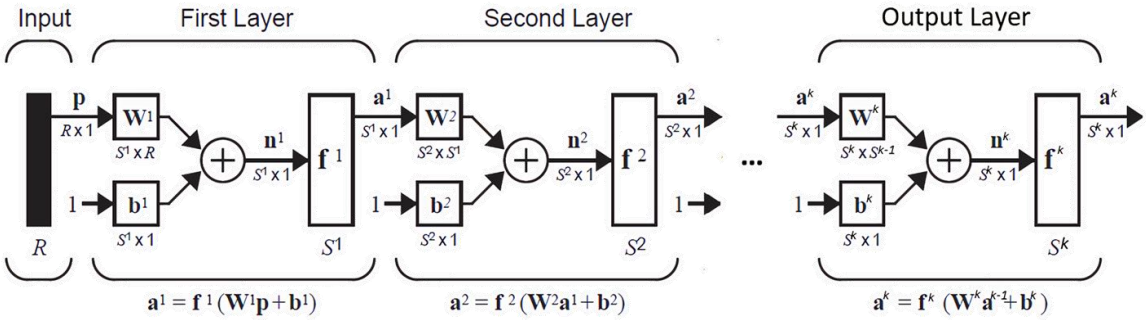
The basic component of a neural network, or a single-input neuron, is shown in Figure 1. The model takes single input *p* to calculate the output through a function *a* which can be expressed as follow:

$$a = f(Wp + b)$$

where *p* is a single scalar input, *w* is the weight, 1 is the other input, *b* is bias. Function *a* above is called the transfer function to produce outputs and can be adjusted by the neural network designer, bias *b* has a constant input of 1, *w* and *b* are both adjustable scalar parameters by some learning rule so that the neuron input and output relationship meets specific goals.

Based on this single neuron component from Figure 1, a simplified typical multi-layer neural network architecture with multiple inputs (*R*), multiple neurons (*S*) and multiple layers (*k*) is illustrated in Figure 2. In this design, an *R* multiple-input neuron has individual inputs p_1, p_2, …, p_R and each is weighted by corresponding elements $w_{1,1}$, $w_{1,2}$, …, $w_{1,R}$ of the weight matrix *W*. In Figure 2, the input vector *p* is represented by the solid vertical bar at the left. The dimensions of *p* are displayed as *R* x 1 indicating the input is a single vector of *R* elements. Each of the *R* inputs is connected to each of the *S* neurons so the weight matrix has a dimension of *S* x *R*. The outputs of layers one and two are the inputs for layers two and three, and so on. There are k^{th} layers in this architecture, the last layer or the output layer has $R = S^k$ inputs, S^k neurons, and an S^k x S^{k-1} weight matrix W^k.

Figure 2 Illustration for a typical multi-input, multi-neuron, and multi-layer neural network architecture



The outputs on the last layer (the output layer in Figure 2) are calculated by the transfer function a^k, which is expressed as follow:

$$a^k = f^k(W^k a^{k-1} + b^k)$$

where *W* is the weight matrix, *a* is the output from previous layer, b is bias vector, and *k* is the number of layers, a^{k-1} is the S^k x 1 input vector on the output layer (with S^{k-1} being the number of neurons on the second last layer). Multilayer networks are more powerful than single-layer networks.

2.2.2 Major Types of Artificial Neural Networks

Artificial neural network and artificial intelligence are developing rapidly with significant progress in the recent decades (Yuchen Wu, 2018). Since the first neural networks model created by McCulloch and Pitts in 1943 (Warren McCulloch, 1943), there have been hundreds of different ANN models (Macukow, 2016). The differences in ANN models include functions, accepted values, topology, and the learning algorithms. Major types of ANNs, each is designed for specific tasks in different ways, include Feedforward Neural Networks (FNNs), Recurrent Neural Networks (RNNs) (Carpenter, 1987), Convolutional Neural Networks (CNNs), Long Short-Term Memory Networks (LSTMs) (Schmidhuber, 1997), Generative Adversarial Networks (GANs), and Radial Basis Function Networks (RBFNs).

- FNNs are simpler type of neural network where data and information flows in one direction from input layer to output layer.
- RNNs are designed for sequential data processing, where the output at each step depends not only on the current input but also on previous inputs in the sequence. They are commonly used for tasks like natural language processing (NLP), time series analysis, and speech recognition.
- CNNs are designed for processing grid-like data such as images. They use convolutional layers to detect patterns and features in the input data, making them highly effective for tasks like image recognition and object detection.
- LSTMs are a type of RNN designed to address the vanishing gradient problem and handle long-term dependencies in sequential data. They are particularly effective for tasks that require capturing long-term patterns and dependencies, such as machine translation, sentiment analysis, and time-series modeling.
- GANs consist of two neural networks, the generator and the discriminator, that are trained together in a competitive setting to create new data from a given training dataset. The generator creates new data samples, while the discriminator distinguishes between real and generated samples. GANs are typically used for image generation and data augmentation.
- RBFNs use radial basis functions as activation functions and are distinguished from other neural networks due to their universal approximation and faster learning speed.

2.2.3 Using Deep Learning Neural Networks for Tabular Data

Deep neural networks models have shown excellent performance and especially when processing complex data such as image, text and sound. However, their adaptation to tabular data tasks remains highly challenging (Vadim Borisov, 2022). The datasets of this project are 2-dimensional tabular data and the purpose of our study is to compare model performance between classic models and deep learning models using neural networks with synthetic data generation and other techniques.

Gorishniy et al. performed a review of major deep learning models for tabular data and concluded that a ResNet-like architecture and a simple adaptation of the Transformer architecture outperform other NN models, but there is no sufficient evidence that neural networks are better than classical models such as gradient boosting decision trees (Y. Gorishniy, 2021). Similarly, Shwartz-Ziv and Armon published a study rigorously comparing a number of neural network models – including TabNet, NODE, and Net-DNF – with XGBoost on various datasets. Their conclusion was that XGBoost outperforms those deep learning models across datasets they used, and they also demonstrated XGBoost requires much less tuning (R. Shwartz-Ziv, 2021). The authors also noted that an ensemble of neural network models and XGBoost performs the best and better than XGBoost alone.

Some studies noted the fact that it is sometimes unclear why neural network models cannot achieve the same level of predictive quality as in other domains such as image classification and natural language processing. Major issues identified by related work include: 1) low quality training data due to missing values, inclusion of outliers, erroneous or inconsistent data, and small data size (A. Sanchez-Morales, 2020) and (Veeramachaneni, 2018); 2) missing, complex or irregular spatial dependencies. There is little spatial correlation between the features in a tabular dataset or the dependencies are rather complex or irregular. The structure and relationships between features will be learned from scratch when training tabular data using neural networks, thus the inductive biases used for homogeneous data, such as Convolutional Neural Networks (CNNs), are unsuitable for modelling tabular data type (Y. Zhu, 2021); 3) dependency on preprocessing. Some studies indicate that tabular

data and deep neural networks performance may strongly depend on selected preprocessing strategy and popular techniques to process categorical features such as one-hot encoding or ordinal encoding methods can lead to a very sparse feature matrix or introduce a synthetic ordering of previously unordered values, therefore very challenging (Khoshgoftaar, 2020). Also preprocessing methods for tabular data may cause information loss leading to reduced predictive performance (E. Fitkov-Norris, 2012); 4) importance of single features. In contrast to deep neural networks, decision-tree algorithms can handle varying feature importance exceptionally well by selecting single features with a threshold and ignoring the rest, while in a typical deep neural network prediction of class requires a coordinated change in many features (Segal, 2018).

Lastly, some studies landed on more promising outlook for using neural networks for tabular data and a hybrid of classical gradient boosting and deep neural network methods may have better performance. While it was concluded that Net-DNF do not consistently beat XGBoost, their results indicate that Net-DNF performance score is not far behind gradient boosting of decision trees. Therefore, Net-DNF offers a meaningful step toward effective usability of processing tabular data with neural networks (L. Katzir, 2021). A hybrid approach is also recommended by Popov et al., in which the authors introduce Neural Oblivious Decision Ensembles (NODE), a new deep learning architecture designed to work with any tabular data. The proposed NODE architecture benefiting from end-to-end gradient-based optimization and the power of multi-layer hierarchical representation learning (S. Popov, 2019). Similarly, a proposed hybrid methods of using gradient boosting and deep neural networks named SAINT performed attention over rows and columns for a tabular dataset, and outperform other deep learning models and gradient boosting models. An enhanced embedding method and a new contrastive self-supervised pre-training method for scarce target labels were used. (G. Somepalli, 2021). Clements, et al. created a novel approach using deep recurrent and causal convolution-based neural networks to address credit risk monitoring with tabular financial data. The deep neural network models outperformed the benchmark non-sequential tree-based model, achieving significant financial savings and earlier detection of credit risk (J. M. Clements, 2020).

2.3 Bayesian Networks and Data Synthesizer

A Bayesian network is a graphical model of the joint probability distribution for a set of variables. The attributes are called *nodes* in the graph, and a conditional relationship between any two attributes is represented as an *edge* between the two nodes. Nodes and edges construct a Bayesian network, and multiple Bayesian networks can be averaged to form a Bayesian model (Young, Graham, & Penny, 2009). Bayesian networks are typically used to draw probabilistic inference about one attribute in the network given the values of other attributes, and therefore are suitable to be used for missing data imputation (Di Zio, Scanu, Coppola, Luzi, & Ponti, 2004) as well as synthetic data generation (Kaur, et al., 2021) (Baowaly, Lin, Liu, & Chen, 2019).

Data synthesizer is a tool that takes a dataset as input and generates a structurally and statistically similar synthetic dataset using Bayesian Networks (Ping, Stoyanovich, & Howe, 2017). DataSynthesizer consists of three modules — DataDescriber, DataGenerator and ModelInspector. DataDescriber collects the user-provided information about data, such as data types and correlations between attributes, and produces a data summary, adding noise to the distributions to preserve privacy. DataGenerator samples from the summary computed by DataDescriber and outputs synthetic data. ModelInspector provides statistics and plots for the users to visually inspect the similarity between the real data and the synthetic data.

To define the correlations between attributes in the dataset, DataSynthesizer can operate in one of three modes. In correlated attribute mode, a differentially private Bayesian network (Zhang, Cormode, Procopiuc, Strivastava, & Xiao, 2014) is used to capture the correlation structure between attributes, then draw samples from this model to construct the result dataset. Independent attribute mode can be used when there is insufficient data to derive a reasonable correlated model. In this mode, a histogram is created for each attribute, noise is added to the histogram to achieve differential privacy, and then samples are drawn for each attribute. Finally, for cases of extremely sensitive data, one can use random mode that simply generates type-consistent random values for each attribute (Ping, Stoyanovich, & Howe, 2017).

In the current research, we used correlated attribute mode as factors that can help to predict blood transfusion are often correlated. When correlated attribute mode is chosen, DataDescriber runs the GreedyBayes algorithm to construct Bayesian networks (BN) to model correlated attributes (see Table 1).

Table 1 Algorithm for GreedayBayes.

Algorithm 1 GreedyBayes(D, A, k)

Require: Dataset D, set of attributes A, maximum number of parents k

Initialize $\mathcal{N} = \emptyset$ and $V = \emptyset$.
Randomly select an attribute X_1 from A.
Add $(X_1, \emptyset)$ to $\mathcal{N}$; add X_1 to V.
for $i = 2, .., |A|$ **do**
  Initialize $\Omega = \emptyset$
  $p = min(k, |V|)$
  for each $X \in A \backslash V$ and each $\Pi \in \binom{V}{p}$ **do**
    Add (X, Π) to Ω
  end for
  Compute mutual information based on D for all pairs in Ω.
  Select (X_i, Π_i) from Ω with maximal mutual information.
  Add (X_i, Π_i) to $\mathcal{N}$.
end for
return $\mathcal{N}$

In the GreedyBayes algorithm, a Bayesian network N is constructed from input dataset D, attributes A, and the maximum number of parents node k, which defaults to 4. V is the set of visited attributes, and Π is a subset of V that will become parents of node X if added to N. Which attributes Π are selected as parents of X is determined greedily by maximizing mutual information (X , Π). The Bayesian networks constructed in this algorithm gives the sampling order for generating attribute values. When constructing noisy conditioned distributions, Lap(4(d−k)) is injected to preserve privacy, where d is the number of attributes, k is the maximum number of parents of a node, and n is the number of tuples in the input dataset.

3 Solution and Methodology

3.1 Data Source and Data Preprocessing

The data was downloaded from the Participant Use Data File (PUF) on the American College of Surgeons National Surgical Quality Improvement Program (ACS NSQIP). In the current research, we focus on the data from 2015 to 2022, which has a total of 13,534 observations and 296 variables across eight datasets.

First, we built a data preprocessing pipeline to clean the raw data, including imputation (mean for numeric variables and most frequent values for categorical variables), standardization, and encoding (e.g., one-hot encoding for categorical variables). Secondly, preprocessed data was reshaped before entering each neural network. Among the 296 variables, 41 features were identified as most relevant to the current study. The target variable is Occurrences Bleeding Transfusions, which is a binary variable predicting whether the patient needs blood transfusion during or after surgery. The target can be further categorized into intraoperative vs. postoperative vs. no transfusion, therefore can be transformed into a 3-class variable when needed. With different analysis strategies, these features were entered into our models to predict the target variable, and we compared the performance with each other as well as with the benchmarks from previous research.

3.2 Exploratory Data Analysis (EDA)

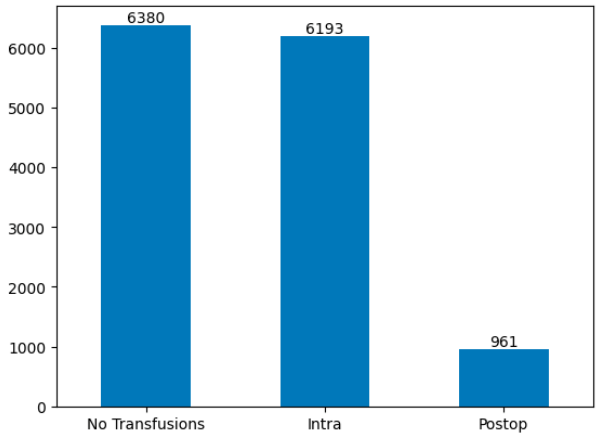
Among the 13,534 patients in the eight-year combined dataset, nearly 80% are male. The mean age is 65.73 with a standard deviation of 9.82. As for ethnicity composition (see Table 1), nearly half of the patients are white (48%) though over a third did not report their ethnicity (44%). Body Mass Index (BMI) were calculated based on HEIGHT and WEIGHT, indicating the signs of overweight with a mean BMI of 29.26 and a standard deviation of 5.76.

Table 1 Ethnicity Composition of CABG Patients.

Ethnicity	Counts
White	6,488
Unknown/Not Reported	5,918
Black or African American	606
Asian	381
Some Other Race	60
American Indian or Alaska Native	38
Native Hawaiian or Pacific Islander	36
Native Hawaiian or Other Pacific Islander	7

Among all CABG patients, around half of the patients had blood transfusion (52.8%) and the other half did not (see Figure 1), therefore the binary target variable Bleeding Occurrence is balanced. If further broken down into intra- vs. postop-blood transfusion, we can see that most of the patients who received blood transfusion had it *during* the surgery (86.5%) and only 13.5% had blood transfusion *after* the surgery.

Figure 3 Bleeding Occurrence Breakdown.



3.3 Analysis Strategy

In this study, we aimed to employ two deep neural networks – Fully-Connected Neural Networks (FNN) and Convolutional Neural Networks (CNN) to predict the need for perioperative blood transfusions for CABG patients. Additionally, we used two approaches to generate synthetic data to train these neural networks – DataSynthesizer and REaLTabFormer (Realistic Relational and Tabular Data using Transformers). Data Synthesizer is based off Bayesian Networks, which are probabilistic graphical models that represent probabilistic relationship between variables. While REaLTabFormer uses a sequence-to-sequence (Seq2Seq) model for generating synthetic relational datasets and uses GPT-2 for non-relational tabular data.

In each type of neural network, we designed different models (e.g., different number of layers, activation functions, loss functions, etc.) and tested them with the original dataset, then we re-ran the models with synthetic datasets from DataSynthesizer and REaLTabFormer and compared the results to see which combination yields the best performance.

4 Results and Discussion

4.1 Fully-Connected Neural Networks (FNNs)

In FNNs, we designed eight models varying in complexity (5-layer vs. 7-layer with more neurons), optimizers (SGD vs. Adam), output activation functions (sigmoid vs. softmax) and their corresponding loss functions (binary cross entropy vs. categorical cross entropy) to see which one(s) makes the best predictions. To evaluate the model performance, we looked at metrics across accuracy, f1 score, area under the curve (AUC), rooted mean squared error (rMSE). The best model(s) are determined jointly by f1 score and accuracy score with rMSE and AUC as supplementary benchmarks.

4.1.1 FNNs with Original Data

Results from FNN with the original dataset were shown in Table 2. Accuracy scores and f1 scores were landed in the range from .68 to .72, with a lowest rMSE of .46 and a highest AUC of .78. The best model was the five-layer design with SGD as optimizer and softmax as output activation function.

Table 2 FNN results with original dataset.

Model	accuracy	f1_score	rMSE	AUC
FNN-5layer-SGD-sigmoid	0.7215	0.6888	0.4794	0.7764
FNN-5layer-Adam-sigmoid	0.7218	0.6961	0.4928	0.7756
FNN-7layer-SGD-sigmoid	0.7174	0.6954	0.4850	0.7741
FNN-7layer-Adam-sigmoid	0.7174	0.6782	0.5735	0.7752
FNN-5layer-SGD-softmax	0.7229	0.7200	0.4643	0.7782
FNN-5layer-Adam-softmax	0.7218	0.7179	0.4975	0.7815
FNN-7layer-SGD-softmax	0.7181	0.7150	0.4780	0.7780
FNN-7layer-Adam-softmax	0.7185	0.7118	0.5587	0.7766

4.1.2 FNNs with Synthetic Data from REaLTabFormer

The eight models showed slightly improved performance with the synthetic data from REaLTabFormer (see Table 3). The accuracy scores and f1 scores ranged from .72 to .74, with a lowest rMSE of .45 and a highest AUC of .80. The best models, once again, were the 5-layer model with SGD using either sigmoid or softmax function.

Table 3 FNN results with synthetic dataset from REaLTabFormer.

Model	accuracy	f1_score	rMSE	AUC
FNN-5layer-SGD-sigmoid	0.7362	0.7420	0.4789	0.8006
FNN-5layer-Adam-sigmoid	0.7359	0.7394	0.4650	0.8007
FNN-7layer-SGD-sigmoid	0.7303	0.7392	0.4819	0.7939
FNN-7layer-Adam-sigmoid	0.7351	0.7315	0.5577	0.7972
FNN-5layer-SGD-softmax	0.7381	0.7380	0.4506	0.8008
FNN-5layer-Adam-softmax	0.7355	0.7355	0.4698	0.7999
FNN-7layer-SGD-softmax	0.7362	0.7362	0.4787	0.7995
FNN-7layer-Adam-softmax	0.7303	0.7286	0.5575	0.7934

4.1.3 FNNs with Synthetic Data from DataSynthesizer

Finally, results from FNN with the synthetic data from DataSynthesizer showed significantly improved performance (see Table 4). The accuracy scores and f1 scores ranged from .86 to .87, with a lowest rMSE of .42 and a highest AUC of .94. The best models were the 5-layer SGD model with sigmoid function and the 7-layer model SGD model with softmax function. However, although the 5-layer SGD model with sigmoid function had high accuracy and f1 score, its rMSE was the highest across all models across three datasets (rMSE = .67). An interesting observation was found with DataSynthesizer synthetic data that rMSE were relatively high with SGD optimizer across the board (rMSE =.55~.67) compared with same design with Adam optimizer (rMSE =.42~.45).

Table 4 FNN results with synthetic dataset from DataSynthesizer.

Model	accuracy	f1_score	rMSE	AUC
FNN-5layer-SGD-sigmoid	0.8718	0.8776	0.6790	0.9386
FNN-5layer-Adam-sigmoid	0.8689	0.8772	0.4340	0.9474
FNN-7layer-SGD-sigmoid	0.8685	0.8774	0.5853	0.9358
FNN-7layer-Adam-sigmoid	0.8626	0.8712	0.4222	0.9456
FNN-5layer-SGD-softmax	0.8751	0.8749	0.6119	0.9421
FNN-5layer-Adam-softmax	0.8696	0.8696	0.4286	0.9495
FNN-7layer-SGD-softmax	0.8762	0.8762	0.5596	0.9401
FNN-7layer-Adam-softmax	0.8670	0.8667	0.4513	0.9474

4.2 Convolutional Neural Networks (CNNs)

While CNN is more suitable for handling image data, we also trained our preprocessed datasets with CNN just for comparison. We first transformed the 2-dimensional tabular dataset to 4-dimensional to meet the data format required by CNN. Two datasets derived from different synthetic data generation techniques (REaLTabFormer and DataSynthesizer) were used to run the CNN model and the accuracy was far lower than the FNN models (see Table 5). This results echo the findings and conclusion from related work done by others and mentioned in the earlier literature review section (Y. Gorishniy, 2021, R. Shwartz-Ziv, 2021, Y.Zhu, 2021)

Table 5 CNN results with across different preprocessed datasets.

Model	dataset	accuracy
CNN-2D-2layer-noPooling-ReLU	synthetic dataset 2015-2022 from REaLTabFormer	0.6681
	synthetic dataset 2015-2022 from DataSynthesizer	0.5785

5 Conclusion

The current research aimed to employ modern techniques to develop models that can best predict the need for blood transfusions among CABG patients. In some cases, deep neural networks combined with data synthesis techniques have shown to significantly improve model performance. Especially in FNNs, regardless of model complexity and design, models trained with synthetic data generated from DataSynthesizer had best performance across the board, with f1 score and accuracy ranged from .86 to .87, with a lowest rMSE of .42 and a highest AUC of .94. Future research should look into different methodologies to generate synthetic data for training and developing models, both tree-based models and deep neural networks, that can help inform guidelines for major high-risk surgeries.

References

A. Sanchez-Morales, J.-L. S.-G.-A.-G.-V. (2020). Improving deep learning performance with missing values via deletion and compensation. *Neural Computing and Applications*, vol. 32, no. 17, pp. 13 233–13 244.

Baowaly, M., Lin, C., Liu, C., & Chen, K. (2019, March). Synthesizing electronic health records using improved generative adversarial networks. *Journal of the American Medical Informatics Association, 26*(3), 228-241.

Carpenter, G. G. (1987). A massively parallel architecture for a self-organizing neural pattern recognition machine. *Neural Networks and Natural Intelligence*, 37, 54-115. DOI: https://doi.org/10.7551/mitpress/4934.003.0008.

Di Zio, M., Scanu, M., Coppola, L., Luzi, O., & Ponti, A. (2004). Bayesian networks for imputation. *Journal of the Royal Statistical Society, 167*, 309-322.

E. Fitkov-Norris, S. V. (2012). Evaluating the impact of categorical data encoding and scaling on neural network classification performance: the case of repeat consumption of identical cultural goods. *International Conference on Engineering Applications of Neural Networks*, pp. 343–352.

G. Somepalli, M. G. (2021). SAINT: Improved neural networks for tabular data via row attention and contrastive pre-training. arXiv preprint arXiv:2106.01342.

Gao, Y., Liu, X., Wang, L., Wang, S., Yu, Y., Ding, Y., . . . Ao, H. (2022, July). Machine learning algorithms to predict major bleeding after isolated coronary artery bypass grafting. *Frontiers in Cardiovascular Medicine, 9*, doi: 10.3389/fcvm.2022.881881.

Horvath, K., Acker, M., Chang, H., & Bagiella et al, E. (2013, June). Blood transfusion and infection after cardiac surgery. *The Annals of Thoracic Surgery, 95*(6), 2194-2201.

J. M. Clements, D. X. (2020). Sequential deep learning for credit risk monitoring with tabular financial data. arXiv preprint arXiv:2012.15330.

Kaur, D., Sobiesk, M., Patil, S., Liu, J., Bhagat, P., Gupta, A., & Markuzon, N. (2021, March). Application of Bayesian networks to generate synthetic health data. *Journal of the American Medical Informatics Association, 28*(4), 801-811. doi: 10.1093/jamia/ocaa303.

Khoshgoftaar, J. T. (2020). “Survey on categorical data for neural networks. *Journal of Big Data*, vol. 7, pp. 1–41.

L. Katzir, G. E.-Y. (2021). Net-DNF: Effective deep modeling of tabular data. *International Conference on Learning.*

Li, Q., Lv, H., Chen, Y., Shen, J., Shi, J., & Zhou, C. (2024, April). Development and validation of a machine learning prediction model for perioperative red blood cell transfusions in cardiac surgery. *International Journal of Medical Informatics, 184*, 105343.

Macukow, B. (2016). Neural Networks – State of Art, Brief History, Basic Models and Architecture. *Computer Information Systems and Industrial Management*, vol 9842. https://doi.org/10.1007/978-3-319-45378-1_1.

Martin T. Hagan, H. B. (2014). *Neural Network Design (2nd Edition).*

Mufarrih, S., Mahmood, F., Qureshi, N., Yunus, R., & Matyal et al, R. (2023, Mar). Timing of blood transfusions and 30-day patient outcomes after coronary artery bypass graft surgery. *Journal of Cardiothoracic and Vascular Anesthesia, 37*(3), 382-391, doi: 10.1053/j.jvca.2022.11.029.

Ping, H., Stoyanovich, J., & Howe, B. (2017). DataSynthesizer: Privacy-preserving synthetic datasets. *International Conference on Scientific and Statistical Database Management*, (pp. 1-5).

R. Shwartz-Ziv, A. A. (2021). Tabular Data: Deep Learning is Not All You Need. arXiv preprint arXiv:2106.03253.

S. Popov, S. M. (2019). Neural oblivious decision ensembles for deep learning on tabular data. *arxiv:1909.06312*.

Schmidhuber, S. H. (1997). Long short-term memory. *Neural Computation*, vol. 9, no. 8, pp. 1735–1780.

Segal, I. S. (2018). "Regularization learning networks: deep learning. *Advances in Neural Information Processing Systems*, pp. 1379–1389.

Tschoellitsch, T., Bock, C., Mahecic, T., Hofmann, A., & Meier, J. (2022, Sep). Machine learning-based prediction of massive perioperative allogeneic blood transfusion in cardiac surgery. *European Journal of Anaesthesiology, 39*(9), 766-773, doi: 10.1097/EJA.0000000000001721.

Vadim Borisov, T. L. (2022). Deep Neural Networks and Tabular Data: A Survey. *IEEE.* https://arxiv.org/pdf/2110.01889.

Veeramachaneni, L. X. (2018). Synthesizing Tabular Data using Generative Adversarial Networks. arXiv preprint arXiv:1811.11264.

Warren McCulloch, W. P. (1943). A logical calculus of the ideas immanent in nervous activity. *Bulletin of Mathematical Biology*, 5, 115-133.

Y. Gorishniy, I. R. (2021). Revisiting deep learning models for tabular data. arXiv preprint arXiv:2106.11959.

Y. Zhu, T. B. (2021). Converting tabular data into images. *Scientific Reports*, vol. 11, no. 1, pp. 1–11.

Young, J., Graham, P., & Penny, R. (2009). Using Bayesian networks to create synthetic data. *Journal of Official Statistics, 25*(4), 549-567.

Yuchen Wu, J. F. (2018). Development and Application of Artificial Neural Network. *Wireless Personal Communications: An International Journal, Volume 102, Issue 2*, 1645-1656. Retrieved from https://doi.org/10.1007/s11277-017-5224-x

Zhang, J., Cormode, G., Procopiuc, C., Strivastava, D., & Xiao, X. (2014). PrivBayes: private data release via bayesian networks. *SIGMOD International Conference on Management of Data*, (pp. 1423-1434).