Proceedings of the 2023 International Conference on Machine Learning and Automation
DOI: 10.54254/2755-2721/40/20230659

Generalisation of Feed-Forward Neural Networks and Recurrent Neural Networks

Rui Wang

Guangzhou Foreign Language School, Guangzhou, China

ccox79911@student.napavalley.edu

Abstract. This paper presents an in-depth analysis of Feed-Forward Neural Networks (FNNs) and Recurrent Neural Networks (RNNs), two powerful models in the field of artificial intelligence. Understanding these models and their applications is crucial for harnessing their potential. The study addresses the need to comprehend the unique characteristics and architectures of FNNs and RNNs. These models excel at processing sequential and temporal data, making them indispensable in tasks. Furthermore, the paper emphasises the importance of variables in FNNs and proposes a novel method to rank the importance of independent variables in predicting the output variable. By understanding the relationship between inputs and outputs, valuable insights can be gained into the underlying patterns and mechanisms driving the system being modelled. Additionally, the research explores the impact of initial weights on model performance. Contrary to conventional beliefs, the study provides evidence that neural networks with random weights can achieve competitive performance, particularly in situations with limited training datasets. This finding challenges the traditional notion that careful initialization is necessary for neural networks to perform well. In summary, this paper provides a comprehensive analysis of FNNs and RNNs while highlighting the importance of understanding the relationship between variables and the impact of initial weights on model performance. By shedding light on these crucial aspects, this research contributes to the advancement and effective utilisation of neural networks, paving the way for improved predictions and insights in various domains.

Keywords: FNN, RNN, Artificial Intelligence.

1. Introduction

Over the past few years, there has been a significant breakthrough in the field of artificial intelligence, driven in part by the advancements in neural networks. Among these neural networks, Feed-Forward Neural Networks (FNNs) and Recurrent Neural Networks (RNNs) have emerged as powerful tools for solving complex problems in machine learning and AI. FNNs and RNNs possess distinct characteristics that make them well-suited for different types of data processing tasks.

FNNs serve as the foundational architecture for neural networks, providing the basic structure and framework for information flow. On the other hand, RNNs excel in addressing sequential problems that arise in time-dependent tasks. Their ability to retain and process information from previous steps makes them particularly valuable in scenarios such as natural language processing and speech recognition.

While newer and more advanced neural network models have been developed, FNNs and RNNs remain fundamental components of machine learning. They embody the underlying logic and principles that form the basis for many successful network architectures. Consequently, gaining a thorough understanding of FNNs and RNNs is essential for comprehending their deep connections with other networks and models, and their influence on the remarkable achievements in artificial intelligence.

This paper aims to elucidate the unique characteristics of FNNs and RNNs, as well as their respective applications across various fields. By exploring their underlying architectures, training methods, and real-world use cases, the paper will provide valuable insights into the performance and suitability of FNNs and RNNs in different problem domains. Understanding these networks' strengths and limitations will contribute to a more comprehensive understanding of their capabilities and facilitate informed decision-making when applying neural networks in practical scenarios.

2. The architecture of FNN and RNN

2.1. FNN

A Feedforward Neural Network (FNN) is a basic type of artificial neural network where input layers receive data, which go through intermediate hidden layers to the output layer to output. General structure is shown in Figure 1. "Feedforward" refers to the absence of any cyclic or feedback connections within the network. It consists of multiple layers: an input layer, a hidden layer, and an output layer. Each layer contains nodes called neurons, which perform computations. Neurons receive inputs from the previous layer, apply weighted summation operations on the inputs, add bias terms for flexibility, and then pass the results through an activation function before transmitting them to the next layer, in order to increase the nonlinearity of the model, enabling it to form more complex representations. Information propagates from the input layer to the hidden layers and finally reaches the output layer during the forward propagation process. During training, after predicting the output, a loss function is used to compare it with the actual labels and compute an error value. This error is then backpropagated through the entire network to adjust the weights and biases using optimisation algorithms, aiming to minimise the discrepancy between the predicted values and the ground truth. This process adjusts the parameters of all layers in the network until the desired outcome is achieved.

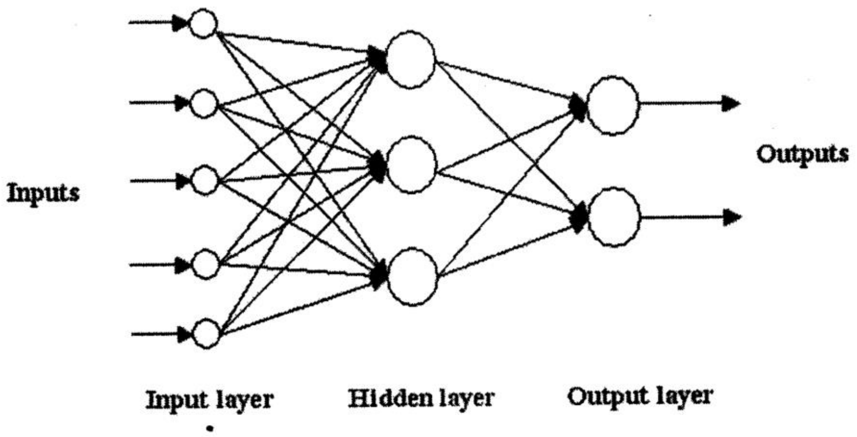

Figure 1. Illustration of FNN [1].

2.2. RNN

Recurrent Neural Network (RNNs) is a variant of artificial neural network that has feedback connections. General structure is shown in Figure 2. Its overall structure is similar to FNN, with the difference that RNNs allow information to circulate within the network, providing a certain degree of memory at each node. This enables RNNs to handle sequential data, such as language, where order is important. The input at each time step is passed to the current layer while providing the hidden state information computed in previous steps.

FNN and RNN have different structure, so they have different characteristics. Due to the unidirectional flow of information, FNNs can fully use resources and perform efficient parallel computations FNNs have a relatively simple architecture, making it easy to understand and interpret the transformation process between inputs and outputs. On the other hand, FNNs cannot capture long-term dependencies in time series data, which may limit their effectiveness in handling sequential data. FNNs also require inputs of fixed sizes and may require padding or truncation for variable-length or differently sized data.

RNNs is able to capture previous information through hidden states, effectively handling time series data. And the architecture of RNNs allows for dynamic computation of the hidden state size at each step, enabling them to naturally handle variable-length or differently sized data. However, Due to the propagation of information through recurrent connections, RNNs often require sequential computation of each step's result, making parallelisation challenging.

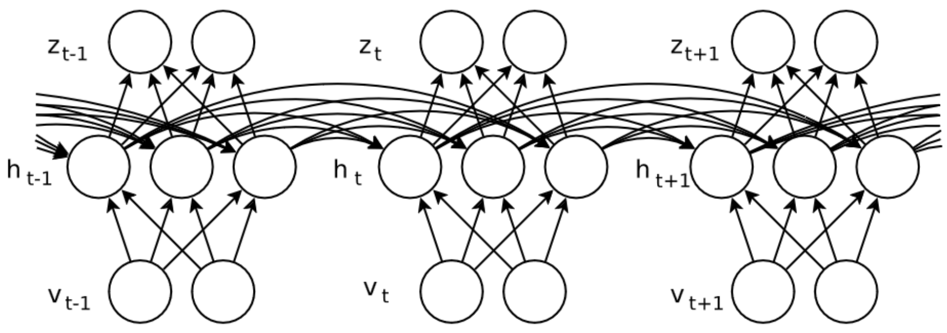

Figure 2. Illustration of RNN [2].

3. The number of hidden layers

In neural networks, hidden layers refer to the layers of nodes or neurons that exist between the input layer and the output layer. The hidden layers perform complex computations and transformations on the input data, extracting relevant features and patterns to facilitate the learning process. The number of hidden layers and the number of neurons within each hidden layer are important architectural parameters that can impact the network's capacity to learn and generalise from the input data. More hidden layers means the model can do more complicated works. However, more hidden layers do not necessarily mean the model has a better performance. [3]

Caching can have a significant impact on performance by reducing data access latency. Therefore, due to caching, the performance of FNN does not always have positive relation to the number of nodes. According to [4], different structure has different caching policies and different performance. Different caching policy and different structures need different numbers of nodes. The study finds that there exists an optimal number of hidden layers for maximising caching efficiency, beyond which increasing the number of hidden layers can lead to diminishing returns or a decrease in performance. This phenomenon can be attributed to the increased computational complexity and potential overfitting associated with excessive hidden layers. Carefully selecting the number of hidden layers in cache systems to reach a balance between computational complexity and caching effectiveness.

4. The contribution of variables in FNNs

The training and application of neural networks are often described as ‘Black Box’, which refers to the characteristic of neural networks where they are perceived as complex systems whose internal workings are not easily interpretable or understandable by humans. So, the inputs have no direct relationship with the output, or at least we cannot understand how they are connected, and we need experiments to find out the relations between them and the contribution of different variables, where Ref. [4] provides a great one.

As stated in Ref. [4], a different approach has been suggested to prioritise the independent variables based on their importance in predicting the output variable. This method utilises the interquartile range of the empirical distribution of network weights derived from training the network. It is also employed to determine the relative significance of predictor variables in real-life scenarios.

Ref. [4] concludes that noise is not a major factor that affects the accuracy, instead, the number of the nodes inside the plays a big role. The networks with fewer nodes are more accurate with the same number of samples. This can be explained because a network with more nodes needs more training samples. What's more, the condition number index, which is used to assess the stability and robustness of a mathematical problem or algorithm, also affects the network a lot. It is found that the more uncertain the samples are, the more inaccurate the model is. This phenomenon can also be explained by our common sense, but this factor has a greater effect than we think.

In the process of model training, many parameters in the network are modified to fit the characteristics of the input samples, including weights, biases, and embeddings. These parameters directly affect the performance of the final model. However, certain study has found that the initial weights in the model are not that important, and they don't need to be trained with high accuracy.

Ref. [5] gives a good demonstration of this perspective. The paper conducted experiments on various benchmark datasets and compared the results with networks initialised using deterministic weights. The authors of the paper found that neural networks with random weights can achieve competitive performance compared to networks with deterministic weights. Additionally, they observed that the convergence speed of networks with random weights was slower initially but improved as the training went on. The study concluded that initialising neural networks with random weights can be a viable approach, especially when the training dataset is limited.

5. Sequentially of RNN

As the paper mentioned before, RNNs have a similar structure to FNNs, but they backpropagate to store some data to keep data in sequence to train the model with order. Technically, RNN can be used to train any kind of data no matter how complex it is as long as the model is huge enough to store enough information. However, while training a model in real life, the performance of computers does not allow them to keep these data for such a long time, A phenomenon like amnesiac often occurs in model training, so different solutions are proposed to help computers store data with a higher efficiency.

Ref. [6] uses long short-term memory-network (LSTM) to help networks reading. Ref. [6] uses long short-term memory architecture [7], which controls the data stored in memory by calculating different statuses at different times and different situations to decide whether to keep it or free it. Long short-term memory has input gates that decide whether data is going to be calculated in the current calculation, forget gates that decide whether data is going to be forgotten, and output gates that decide whether data is going to pass to the next sequence. Instead of rearranging files in memory which consumes some performance of the calculation, Ref. [6] proposed an alternative method – files stored in memory can be called by labels that indicate the address of the file in memory. Passing labels between nodes accelerates the training process.

Another way is using Gated Recurrent Unit (GRU) [8]. GRU receives the data with the time step along with the hidden state from the previous time step as inputs. It has a reset gate that determines how to combine the previous hidden state with the current input. The update gate determines how to incorporate the current time step's information into the hidden state. Although GRU has a simpler structure, according to [9], it also performs well, just like LSTM, way better than the traditional tanh unit.

The alternative method to this Is the attention mechanics [10] which is also applied to transformer models. The first step is input, it takes three inputs: the current state or context, information that the model needs to pay attention to, and the complete sequence of data that the model is processing. Then the attention weights for each input element or token is computed. After computing the attention weights, a softmax function is applied to normalise the weights, ensuring that they sum up to 1. This normalization allows the model to distribute its attention across the input sequence. The attention weights are then used to compute a weighted sum of the corresponding values. The weighted sum of the values produces a context vector, which represents a weighted combination of the information in the input sequence. The context vector is typically combined with the query or the current state of the model using concatenation or element-wise addition.

6. Conclusion

In conclusion, Feed-Forward Neural Networks (FNNs) and Recurrent Neural Networks (RNNs) represent crucial components of machine learning and artificial intelligence. While FNNs stand out in efficient parallel computations and are suitable for tasks that do not require capturing long-term dependencies, RNNs are specifically designed to handle sequential data by leveraging hidden states and recurrent connections. The unique characteristics of each network make them applicable to different problem domains.

Understanding the contribution of variables in FNNs is essential for interpreting the relationship between inputs and outputs. This method based on the interquartile range of network weights provides insights into the relative important of predictor variables. Moreover, studies have shown that initial weights in neural networks do not need to be trained with high accuracy, and random weight initialisation can be a viable approach, particularly in scenarios with limited training data.

In the case of RNNs, different techniques such as Long Short-Term Memory (LSTM), Gated Recurrent Units (GRUs), and attention mechanisms have been developed to handle sequential data efficiently. These approaches address challenges related to memory storage, information flow, and computational efficiency, enabling RNNs to effectively capture dependencies in time series data.

By gaining a comprehensive understanding of FNNs and RNNs, researchers and practitioners can make informed decisions when selecting the appropriate network architecture for specific tasks. Future research efforts should continue to explore advanced variations and combinations of these networks to further improve their performance and extend their applications in various domains. The continuous development of artificial intelligence relies on a solid foundation of understanding the fundamental concepts and capabilities of neural networks.

References

[1] Sazli M. H. Communications Faculty of Sciences University of Ankara Series A2-A3 Physical Sciences and Engineering 2006 50.01
[2] Sutskever I. Training recurrent neural networks. Toronto, ON, Canada: University of Toronto, 2013.
[3] Fedchenko V. Giovanni N. and Bruno R. 2019 46.3 139-142.
[4] Paliwal M. and Usha A. K. Applied Soft Computing 2011 11.4 3690-3696.
[5] Schmidt W. F. Martin A. K. and Robert PW D. International conference on pattern recognition. IEEE Computer Society Press, 1992.
[6] Cheng J. Li D. and Mirella L. arXiv preprint arXiv 2016 1601.06733
[7] Hochreiter S. and Jürgen S. Neural computation 1997 9.8 1735-1780.
[8] Dey R. and Fathi M. S. 2017 IEEE 60th international midwest symposium on circuits and systems (MWSCAS). IEEE, 2017.
[9] Chung J. et al. arXiv preprint arXiv 2014 1412.3555
[10] Niu Z. Guoqiang Z. and Hui Y. Neurocomputing 2021 452 48-62.