

A Statistical-Modelling Approach to Feedforward Neural Network Model Selection

Andrew McInerney*

Kevin Burke†

May 2, 2024

Abstract

Feedforward neural networks (FNNs) can be viewed as non-linear regression models, where covariates enter the model through a combination of weighted summations and non-linear functions. Although these models have some similarities to the approaches used within statistical modelling, the majority of neural network research has been conducted outside of the field of statistics. This has resulted in a lack of statistically-based methodology, and, in particular, there has been little emphasis on model parsimony. Determining the input layer structure is analogous to variable selection, while the structure for the hidden layer relates to model complexity. In practice, neural network model selection is often carried out by comparing models using out-of-sample performance. However, in contrast, the construction of an associated likelihood function opens the door to information-criteria-based variable and architecture selection. A novel model selection method, which performs both input- and hidden-node selection, is proposed using the Bayesian information criterion (BIC) for FNNs. The choice of BIC over out-of-sample performance as the model selection objective function leads to an increased probability of recovering the true model, while parsimoniously achieving favourable out-of-sample performance. Simulation studies are used to evaluate and justify the proposed method, and applications on real data are investigated.

Keywords. Neural networks; Model selection; Variable selection; Information criteria.

*Department of Mathematics and Statistics, University of Limerick; andrew.mcinerney@ul.ie

†Department of Mathematics and Statistics, University of Limerick; kevin.burke@ul.ie

1 Introduction

Neural networks are a popular class of machine-learning models, which pervade modern society through their use in many artificial-intelligence-based systems (LeCun et al., 2015). Their success can be attributed to their predictive performance in an array of complex problems (Abiodun et al., 2018). Recently, neural networks have been used to perform tasks such as natural language processing (Goldberg, 2016), anomaly detection (Pang et al., 2021), and image recognition (Voulodimos et al., 2018). Feedforward neural networks (FNNs), which are a particular type of neural network, can be viewed as non-linear regression models, and have some similarities to statistical modelling approaches (e.g., covariates enter the model through a weighted summation, and the estimation of the weights for an FNN is equivalent to the calculation of a vector-valued statistic) (Ripley, 1994; White, 1989). Despite early interest from the statistical community (White, 1989; Ripley, 1993; Cheng and Titterton, 1994), the majority of neural network research has been conducted outside of the field of statistics (Breiman, 2001; Hooker and Mentch, 2021). Given this, there is a general lack of statistically-based methods, such as model and variable selection, which focus on developing parsimonious models.

Typically, the primary focus when implementing a neural network centres on model predictivity (rather than parsimony); the models are viewed as ‘black-boxes’ whose complexity is not of great concern (Efron, 2020). It is perhaps not surprising, therefore, that there is a tendency for neural networks to be highly over-parameterised, miscalibrated, and unstable (Sun et al., 2022). Nevertheless, FNNs can capture more complex covariate effects than is typical within popular (linear/additive) statistical models. Consequently, there has been renewed interest in merging statistical models and neural networks, for example, in the context of flexible distributional regression (Rügamer et al., 2020) and mixed modelling (Tran et al., 2020). However, statistically-based model selection procedures are required to increase the utility of the FNN within the statistician’s toolbox.

Traditional statistical modelling is concerned with developing parsimonious models, as it is crucial for the efficient estimation of covariate effects and significance testing (Efron, 2020). Indeed, model selection (which includes variable selection) is one of the fundamental problems of statistical modelling (Fisher and Russell, 1922). It involves choosing the “best” model, from a range of candidate models, by trading pure data fit against model complexity (Anderson and Burnham, 2004). As such, there has been a substantial amount of research on model and variable selection (Miller, 2002). As noted by Heinze et al. (2018), typical approaches include significance testing combined with forward selection or backward elimination (or a combination thereof); information criteria such as AIC or BIC (Akaike, 1998; Schwarz, 1978; Anderson and Burnham, 2004); and penalised likelihood such as LASSO (Tibshirani, 1996; Fan and Lv, 2010).

In machine learning, due to the focus on model predictivity, relatively less emphasis is placed on finding a model that strikes a balance between complexity and fit. Looking at FNNs in particular, the number of hidden nodes is usually treated as a tunable hyperparameter (Bishop et al., 1995; Pontes et al., 2016). Input-node selection is not as common,

as the usual consensus when fitting FNNs appears to be similar to the early opinion of Breiman (2001): “the more predictor variables, the more information”. However, there are some approaches in this direction, and a survey of variable selection techniques in machine learning can be found in Chandrashekar and Sahin (2014). Nevertheless, typically, the optimal model is usually determined based on its predictive performance, such as out-of-sample mean squared error, which can be calculated on a validation data set. Unlike an information criteria, out-of-sample performance does not directly take account of model complexity.

When framing an FNN statistically, there are several motivating reasons for a model selection procedure that aims to obtain a parsimonious model. For example, the estimation of parameters in a larger-than-required model results in a loss in model efficiency, which, in turn, leads to less precise estimates. Input-node selection, which is often ignored in the context of neural networks, can provide the practitioner with insights on the importance of covariates. Instead, other feature importance measures are typically used such as the feature attribution methods described in Koenen and Wright (2024). Furthermore, eliminating irrelevant covariates can result in cheaper models by reducing potential costs associated with data collection (e.g. financial, time, energy). In this paper, we take a statistical-modelling view of neural network selection by assuming an underlying (normal) error distribution. Doing so enables us to construct a likelihood function, and, hence, carry out information-criteria-based model selection, such as the BIC (Schwarz, 1978), naturally encapsulating the parsimony in the context of a neural network. More specifically, we propose an algorithm that alternates between selecting the hidden layer complexity and the inputs with the objective of minimizing the BIC. We have found, in practice, that this leads to more parsimonious neural network models than the more usual approach of minimizing out-of-sample error, while also not compromising the out-of-sample performance itself.

The remainder of this paper is structured as follows. In Section 2, we introduce the FNN model while linking it to a normal log-likelihood function. Section 3 motivates and details the proposed model selection procedure. Simulation studies to investigate the performance of the proposed method, and to compare it to other approaches, are given in Section 4. In Section 5, we apply our method to real-data examples. Finally, we conclude in Section 6 with a discussion.

2 Feedforward Neural Network

Let $y = (y_1, y_2, \dots, y_n) \in \mathbb{R}^n$ be the response variable of interest for a regression-based problem, where n represents the number of observations. For the i th observation, $i = 1, \dots, n$, let $x_i = (x_{1i}, x_{2i}, \dots, x_{pi})^T$ be a vector of p covariates—the inputs to the neural network model. We assume a model of the form $y_i = \text{NN}(x_i) + \varepsilon_i$, where ε_i is a random

error that we assume has a $N(0, \sigma^2)$ distribution, and $\text{NN}(\cdot)$ is a neural network,

$$\text{NN}(x_i) = \gamma_0 + \sum_{k=1}^q \gamma_k \phi \left(\sum_{j=0}^p \omega_{jk} x_{ji} \right). \quad (2.1)$$

As we aim to frame FNNs as an alternative to other statistical non-linear regression models (i.e., used on small-to-medium sized tabular data sets relative to the much larger data sets seen more broadly in machine learning), and due to the universal approximation theorem (Cybenko, 1989; Hornik et al., 1989), we are restricting our attention to FNNs with a single-hidden layer. The parameters in Equation 2.1 are as follows: ω_{0k} , the intercept term associated with the k th hidden node; ω_{jk} , the weight that connects the j th input node to the k th hidden node; γ_0 , the intercept term associated with the output node; and γ_k , the weight that connects the k th hidden node to the output node. The function $\phi(\cdot)$ is the activation function for the hidden layer, which is often a logistic function. The number of parameters in the neural network is given by $K = (p+2)q + 1$. A diagram of a neural network architecture with p input nodes and q hidden nodes is shown in Figure 1. In the diagram, $x_0 = 1$, $h_0 = 1$, and $h_k = \phi \left(\sum_{j=0}^p \omega_{jk} x_{ji} \right)$.

Given our assumption that $\varepsilon_i \sim N(0, \sigma^2)$, we then make use of the log-likelihood function

$$\ell(\theta) = -\frac{n}{2} \log(2\pi\sigma^2) - \frac{1}{2\sigma^2} \sum_{i=1}^n (y_i - \text{NN}(x_i))^2, \quad (2.2)$$

where $\theta = (\omega_{01}, \dots, \omega_{p1}, \dots, \omega_{0q}, \dots, \omega_{pq}, \gamma_0, \dots, \gamma_q, \sigma^2)^T$. We maximise this log-likelihood to obtain $\hat{\theta}$ but note that the estimates of the neural network parameters do not depend on the value of σ^2 , i.e., the residual sum of squares, $\sum_{i=1}^n (y_i - \text{NN}(x_i))^2$, can be estimated to obtain the neural network parameters. This is useful since standard neural network software (that minimises the residual sum of squares) such as **nnet** (Ripley and Venables, 2022) can be used to optimise the neural network followed by the estimation of σ^2 in a separate step.

The calculation of a log-likelihood function allows for the use of information criteria when selecting a given model, and in particular, the Bayesian information criterion (BIC) (Schwarz, 1978), $\text{BIC} = -2\ell(\hat{\theta}) + \log(n)(K+1)$, where we have $K+1$ parameters, i.e., the K neural network parameters plus the variance parameter, σ^2 . An attractive property of the BIC is that it is “dimension-consistent”, i.e., the probability of selecting the “true” model approaches one as sample size increases (Anderson and Burnham, 2004). It is important to note that other approaches for the calculation of the degrees of freedom exist (Murata et al., 1994; Ye, 1998), but we find these do not penalise more complex models (with redundancies) heavily enough in the model selection context compared to using K (see Appendix A).

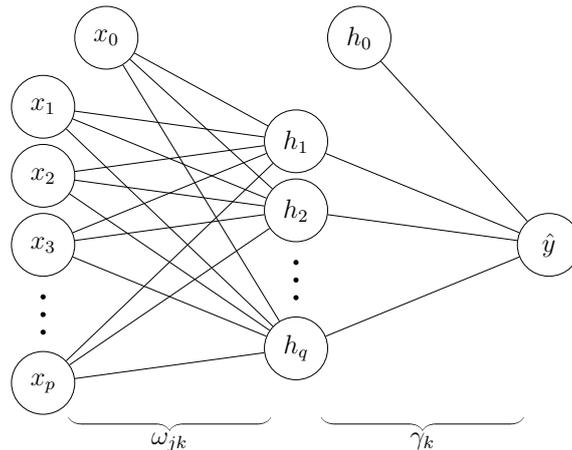


Figure 1: Neural network architecture with p input nodes and q hidden nodes.

3 Model Selection

To begin model selection, a set of candidate models must be considered. For the input layer, we can have up to p_{\max} inputs, where p_{\max} is the maximum number of covariates being considered, and this is often the total number of covariates available in the data under study. The input layer can contain any combination of these p_{\max} inputs. For the hidden layer, we must specify a q_{\max} value, which is the maximum number of hidden nodes to be considered; this controls the maximum level of complexity of the candidate models. We can then have between one and q_{\max} nodes in the hidden layer. From a neural network selection perspective, we aim to select a subset of $p \leq p_{\max}$ covariates to enter the input layer and to build a hidden layer of $q \leq q_{\max}$ nodes to adapt to the required complexity. To carry out these selections, we suggest a statistically-motivated procedure based on minimising the BIC, since it directly penalises complexity and is known to be selection consistent, i.e., BIC minimisation converges to the true model asymptotically. In contrast, and more usually in machine learning applications, one could consider predictive performance, for example, the out-of-sample mean squared error. We will also consider this approach but find that it leads to significantly more complex models than the use of BIC while only marginally improving predictive performance. Whether one is aiming to minimise BIC or out-of-sample mean squared error, multiple initialisations of the neural network (from n_{init} random vectors of parameters) are required to improve the chance of finding a global maximiser of the log-likelihood surface.

3.1 Proposed Approach

We propose a stepwise procedure that starts with a hidden-node selection phase followed by an input-node selection phase. (We find that this ordering leads to improved model selection.) This is, in turn, followed by a fine-tuning phase that alternates between the hidden and input layers for further improvements. The proposed model selection proce-

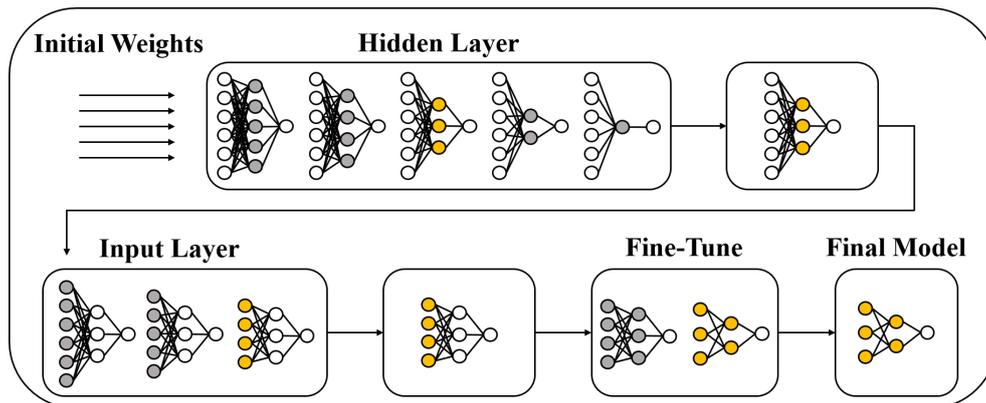


Figure 2: Model selection schematic. Nodes coloured grey are being considered in current phase. Nodes coloured gold represent optimal nodes in that phase to be brought forward to the next phase.

procedure is detailed in Algorithm 4 (which relies on Algorithms 1–3), and a schematic diagram is provided in Figure 2. It is also described at a high level in the following paragraphs.

The procedure (Algorithm 4) is initialised with the full set of input nodes, $\mathcal{X}_{\text{full}}$, the maximum number of hidden nodes being considered, q_{max} , and the number of initialisations, n_{init} , and, as mentioned, starts with a hidden-node selection phase (Algorithm 2 with $Q = \{1, 2, \dots, q_{\text{max}} - 1\}$). For each candidate model in this phase (i.e., models with $q \in \{1, \dots, q_{\text{max}}\}$), the network optimiser is supplied with n_{init} random vectors of initial parameters, the log-likelihood function is maximised at each of these vectors, and the overall maximiser is found (see Algorithm 1). The reason for supplying the neural network with different vectors of initial parameters is due to the complex optimisation surface for neural networks that may contain several local maxima. Thus, the use of a set of initial vectors (rather than just one) aims to increase the chance of finding the global maxima; of course, this cannot be guaranteed as is often the case in more complex statistical models. Once all of the q_{max} candidate models have been fitted, the hidden-node selection phase is concluded by selecting the one whose hidden structure (i.e., number of nodes, q) minimises the BIC.

Once the hidden-node selection phase has concluded, the focus switches to the input layer (Algorithm 3); at this point, there are p_{max} inputs (i.e., the set of input nodes currently included in the model is the set of all input nodes, $\mathcal{X} = \mathcal{X}_{\text{full}}$). For the input-node selection phase, each input node is dropped in turn, with the aim of finding an input whose removal yields a lower BIC; as with the previous phase, random sets of initial parameters are used for each candidate model in the underlying likelihood optimisation. If the removal of a given input node does yield a lower BIC value, then that input node is dropped from the model (and if two or more inputs result in a lower BIC, the one yielding the lower BIC is removed). This is repeated until no covariate, when removed from the model, results in a lower BIC, and, then, the set of included input nodes, \mathcal{X} , is returned. (Thus, in this phase, Algorithm 3 is applied with only the “drop inputs” step

Algorithm 1 Fit Candidate Model

Input: The set of input nodes, \mathcal{X} , the number of hidden nodes, q , and the number of initialisations, n_{init} .

1. Generate n_{init} random initial weight vectors of size $K = (p + 2)q + 1$ with $p = |\mathcal{X}|$ and $|\cdot|$ is the cardinality of a set.
2. Using a neural network optimiser, maximise the log-likelihood function (Equation 2.2) for each initialisation.
3. Select the model with the maximum log-likelihood value as the candidate model.
4. Calculate the associated BIC value.

Output: A fitted neural network with its associated BIC value.

Algorithm 2 Hidden-Node Selection

Input: The set of input nodes currently included in the model, \mathcal{X} , the number of hidden nodes currently in the model, q , the set of hidden-layer structures being considered, Q , and the number of initialisations, n_{init} .

1. For k in Q :

Perform Algorithm 1 with the set \mathcal{X} of input nodes, k hidden nodes, and n_{init} initialisations.

If $\text{BIC}(k) \leq \text{BIC}(q)$:

Set $q = k$.

Output: The number of hidden nodes, q .

and $n_{\text{steps}} = p_{\text{max}}$.)

Both the hidden layer and covariate selection phases are backward elimination procedures. Rather than stopping the algorithm after these two phases, we have found it fruitful to search for an improved model in a neighbourhood of the current “best” model by carrying out some further fine tuning. This is done by considering the addition or removal of one hidden node (Algorithm 2 with $Q = \{q - 1, q + 1\}$), then the further addition or removal of one input node (Algorithm 3 with both the “drop inputs” and “add inputs” steps and $n_{\text{steps}} = 1$), and these two steps are repeated alternately until no further adjustment decreases the BIC (see Step 3 in Algorithm 4). This fine-tuning stage is analogous to stepwise model selection with backward and forward steps. Note that one could apply this alternating stepwise procedure from the offset, but we have found it to be significantly more computationally efficient to focus first on the hidden and input layers (separately and in that order) before moving to the stepwise phase.

The particular order of the model selection steps described above has been chosen in order to have a higher probability in recovering the “true” model, and to have a lower computational cost (see Section 4.1 for a detailed simulation). Note that choosing the *set*

Algorithm 3 Input-Node Selection

Input: The set of all input nodes under consideration, $\mathcal{X}_{\text{full}}$, the set of input nodes currently included in the model, \mathcal{X} , the number of hidden nodes currently in the model, q , the limit on the number of iterations of the repeat step, n_{steps} , the number of initialisations, n_{init} , and this Algorithm covers the possibility of both dropping and adding input variables depending on whether Steps 2(a) and/or 2(b) are applied.

1. Set $i = 0$ and $\mathcal{X}_{\text{new}} = \mathcal{X}$.

2. **Repeat:**

(a) **If drop inputs:**

i. **For c in \mathcal{X} :**

Perform Algorithm 1 with the set $\mathcal{X} \setminus \{c\}$ of input nodes, q hidden nodes, and n_{init} initialisations, where $\mathcal{X} \setminus \{c\}$ is the set \mathcal{X} of input nodes with input node c removed.

If $\text{BIC}(\mathcal{X} \setminus \{c\}) \leq \text{BIC}(\mathcal{X}_{\text{new}})$:

Set $\mathcal{X}_{\text{new}} = \mathcal{X} \setminus \{c\}$.

(b) **If add inputs:**

i. **For c in $\mathcal{X}_{\text{full}} \setminus \mathcal{X}$:**

Perform Algorithm 1 with the set $\mathcal{X} \cup \{c\}$ of input nodes, q hidden nodes, and n_{init} initialisations, where $\mathcal{X} \cup \{c\}$ is the set \mathcal{X} of input nodes with input node c added.

If $\text{BIC}(\mathcal{X} \cup \{c\}) \leq \text{BIC}(\mathcal{X}_{\text{new}})$:

Set $\mathcal{X}_{\text{new}} = \mathcal{X} \cup \{c\}$.

(c) **If $\mathcal{X} \neq \mathcal{X}_{\text{new}}$:**

Set $\mathcal{X} = \mathcal{X}_{\text{new}}$ and $i = i + 1$.

Else:

End repeat.

(d) **If $i \geq n_{\text{steps}}$:**

End repeat.

Output: The set of included input nodes, \mathcal{X} .

of input nodes requires a more extensive search than choosing the *number* of hidden nodes. There are more candidate structures for the input layer as you can have any combination of the nodes. Therefore, it is recommended to perform hidden-node selection first, to eliminate any redundant hidden nodes and decrease the number of parameters in the model, before performing input-node selection.

Algorithm 4 Model Selection

Input: The set of all input nodes, $\mathcal{X}_{\text{full}} = \{x_1, x_2, \dots, x_{p_{\text{max}}}\}$, the maximum number of hidden nodes to be considered, q_{max} , and the number of initialisations, n_{init} .

1. Hidden-Node Selection:

Perform Algorithm 2 with $\mathcal{X} = \mathcal{X}_{\text{full}}$, $Q = \{1, 2, \dots, q_{\text{max}} - 1\}$, $q = q_{\text{max}}$, and $n_{\text{init}} = n_{\text{init}}$.

2. Input-Node Selection:

Perform Algorithm 3 with $\mathcal{X}_{\text{full}} = \mathcal{X}_{\text{full}}$, $\mathcal{X} = \mathcal{X}_{\text{full}}$, $q = q$, $n_{\text{steps}} = p_{\text{max}}$, and $n_{\text{init}} = n_{\text{init}}$, applying only the “drop inputs” step.

3. Fine Tuning:

- **Repeat:**

- (a) **Hidden Layer:**

- Perform Algorithm 2 with $\mathcal{X} = \mathcal{X}$, $Q = \{q - 1, q + 1\}$, $q = q$, and $n_{\text{init}} = n_{\text{init}}$.

- If Step 3(a) did not update the value of q :**

- End repeat.*

- (b) **Input Layer:**

- Perform Algorithm 3 with $\mathcal{X}_{\text{full}} = \{x_1, x_2, \dots, x_{p_{\text{max}}}\}$, $\mathcal{X} = \mathcal{X}$, $q = q$, $n_{\text{steps}} = 1$, and $n_{\text{init}} = n_{\text{init}}$, applying both the “drop inputs” and “add inputs” steps.

- If Step 3(b) did not update the value of \mathcal{X} :**

- End repeat.*

Output: The set of included input nodes, \mathcal{X} , and the number of hidden nodes, q .

* Note: The fine-tuning phase stops if either Step 3(a) or Step 3(b) does not find an improvement. This is to avoid either input-node or hidden-node selection being repeated under conditions previously considered.

4 Simulation Studies

In order to justify and evaluate the proposed model selection approach, three simulation studies are used:

- **Simulation 1 (Section 4.1):** In our first simulation study, we investigate the effect of the ordering of the model selection steps to justify the procedure. This includes the effect of performing input-node and hidden-node selection phases first, the improvement of including a stepwise fine-tuning step, and the performance of a procedure that only carries out iterative stepwise steps (i.e., fine tuning from the

offset).

- **Simulation 2 (Section 4.2):** The second simulation study compares the performance of using the BIC as the model selection objective function versus using AIC or out-of-sample mean squared error (OOS).
- **Simulation 3 (Section 4.3):** The third simulation study investigates the performance of the proposed model selection procedure in the case where the true data-generating process is not a neural network, but, rather is that of a linear-type regression model (albeit with non-linear and interaction terms). Here, we compare the performance of our procedure against classical linear-regression stepwise selection.

In the first two simulation studies, the response is generated from an FNN with known “true” architecture. The weights are generated so that there are three important inputs, x_1, x_2, x_3 , with non-zero weights, and ten unimportant inputs, x_4, \dots, x_{13} , with zero weights. All input variables are independent and generated from a standard normal distribution and the error variance is 0.7 (but the results are similar when the inputs are correlated as shown in Appendix B). The “true” hidden layer consists of $q = 3$ hidden nodes, while we set our procedure to consider a maximum of $q_{\max} = 10$ hidden nodes. The weights of the neural network are held constant over all repetitions and are given by $(\omega_{01} = 1.40, \omega_{11} = 4.35, \omega_{21} = 3.22, \omega_{31} = -2.43, \omega_{02} = -2.89, \omega_{12} = 4.28, \omega_{22} = -3.27, \omega_{32} = -2.30, \omega_{03} = -1.90, \omega_{13} = 4.49, \omega_{23} = 3.24, \omega_{33} = 2.46, \gamma_0 = 2.98, \gamma_1 = 2.37, \gamma_2 = 2.37, \gamma_3 = 2.47)^T$. The metrics calculated to evaluate the performance of the model selection approach are the true negative rate (TNR) for the input nodes (i.e., the proportion of input nodes with true zero weights that are *correctly* dropped from the model), the false discovery rate (FDR) for the input nodes (i.e., the proportion of input nodes with true zero weights that are *incorrectly* included in the model), the average number of hidden nodes selected (\bar{q}), the probability of choosing the correct set of inputs (PI), the probability of choosing the correct number of hidden nodes (PH), and the probability of choosing the overall true model (PT). (All probabilities refer to the proportion of correct results from the 1,000 simulation replicates.) In all simulation studies, we vary the sample size $n \in \{250, 500, 1000\}$ and carry out 1,000 replicates. Our proposed model selection approach is implemented in our publicly available R package `selectnn` (McInerney and Burke, 2022). The neural network function used is `nnet`, which is available from the R package of the same name (Ripley and Venables, 2022). (Note that we do not use a weight decay penalty when fitting the models, i.e., we set `decay = 0` within the `nnet` function.)

4.1 Simulation 1: Model Selection Approach

This simulation study aims to justify the approach of the proposed model selection procedure, i.e., a hidden-node phase, followed by an input-node phase, followed by a fine-tuning phase; here, we label this approach as H-I-F. Some other possibilities would be: to start

with the input-node phase (I-H-F), to stop the procedure without fine tuning (either H-I or I-H), or to only carry out fine-tuning from the beginning (F). Descriptions of the considered model selection approaches are as follows (the proposed approach is highlighted in bold; round brackets indicate the reordering of the steps in Algorithm 4 required to achieve the approach):

- H-I: Hidden-node selection phase, followed by input-node selection phase (Step 1 → Step 2).
- I-H: Input-node selection phase, followed by hidden-node selection phase (Step 2 → Step 1).
- **H-I-F: Hidden-node selection phase, followed by input-node selection phase, and then a fine-tuning phase (Step 1 → Step 2 → Step 3).**
- I-H-F: Input-node selection phase, followed by hidden-node selection phase, and then a fine-tuning phase (Step 2 → Step 1 → Step 3).
- F: Fine-tuning phase only (Step 3).

The objective function used for model selection is BIC, and each approach has $n_{\text{init}} = 5$ initial vectors for the optimisation procedure. (The choice of objective function and the effect of n_{init} are investigated in Section 4.2 and Appendix C, respectively.) The results of the simulation study are shown in Table 1. Boxplots for TNR for the inputs and q for all approaches are displayed in Figure 3 and Figure 4, respectively. The true-positive rate is not shown as it is one for all methods.

Looking at the the model selection metrics, it is clear that the proposed H-I-F approach performs well, both in terms of selecting the correct set of input nodes and selecting the correct number of hidden nodes. Furthermore, the TNR is high, the FDR is low, and, as expected, we see that performance improves across all metrics with increasing sample size. From the results in Table 1, and from Figures 3 and 4, it is clear that the H-I-F approach performs best at recovering the true model structure.

Comparing the methods without the fine-tuning stage in the boxplots, and looking at layerwise selection, the probability of selecting the correct structure is increased when that layer is selected in the second phase, e.g., input-node selection is best when it comes second (see H-I versus I-H in Figure 3). This suggests a relationship between the structure of the input and hidden layers (the probability of correctly selecting the structure of one layer increases when the other layer is more correctly specified). This is investigated further in Appendix D. Therefore, H-I is likely better than I-H due to input-node selection being a more difficult task than hidden-node selection (determining the optimal *set* of input nodes versus the optimal *number* of hidden nodes), and, hence, it is favourable to perform it after hidden-node selection (given the number of hidden nodes is not substantially larger than the number of input nodes). This relationship between the structure of both layers can be handled by incorporating a fine-tuning phase after both the H and I phases are completed. Recall that the aim of fine tuning is to search for an improved solution

Table 1: Simulation 1: model selection metrics.

n	Method	Time (s)	Input layer			Hidden layer		PT
			TNR	FDR	PI	\bar{q} (3)	PH	
250	H-I	13	0.78	0.23	0.59	2.29	0.18	0.10
	I-H	50	0.25	0.70	0.01	2.85	0.44	0.01
	H-I-F	14	0.87	0.15	0.72	2.66	0.54	0.43
	I-H-F	53	0.46	0.61	0.03	2.87	0.50	0.03
	F	116	0.77	0.29	0.47	8.58	0.13	0.12
500	H-I	32	0.90	0.10	0.83	3.47	0.53	0.50
	I-H	100	0.64	0.36	0.42	3.14	0.87	0.40
	H-I-F	36	0.96	0.05	0.90	3.05	0.95	0.85
	I-H-F	103	0.72	0.32	0.46	3.08	0.92	0.43
	F	82	0.97	0.04	0.89	3.17	0.90	0.82
1000	H-I	53	1.00	0.00	0.99	3.02	0.98	0.97
	I-H	186	0.87	0.14	0.78	3.00	1.00	0.77
	H-I-F	53	1.00	0.00	0.99	3.00	1.00	0.99
	I-H-F	189	0.88	0.14	0.77	3.01	0.99	0.76
	F	169	0.99	0.02	0.97	3.04	0.99	0.96

Time (s), median time to completion in seconds (carried out on an Intel[®] Core[™] i5-10210U Processor). Best values for a given sample size are highlighted in **bold**.

in a neighbourhood of the current solution, where both H and I steps are carried out alternately (and include both backward and forward selections). Indeed, we see that the addition of the fine-tuning phase improves on H-I in the smaller sample sizes (in large part due to improved hidden-layer selection), but its addition does not greatly improve on I-H. Moreover, a boxplot for the computational time for each approach is provided in Appendix E, and the addition of fine tuning only marginally adds to the computational expense. Overall, H-I-F is significantly better than I-H-F both in terms of computational expense and model selection. One may also consider only carrying out fine-tuning steps from the offset, which we denote by F. However, this does not perform as well as H-I-F at the smallest sample size and is more computationally demanding. From the above, the H-I-F approach is what we suggest as it leads to good model selection performance while also being the most computationally efficient approach.

4.2 Simulation 2: Model Selection Objective Function

This simulation study aims to determine the performance of using different objective functions when carrying out model selection. In particular, it aims to determine whether the use of an information criterion can improve the ability for the model selection procedure to recover the true model; this is compared to the far more common approach in neural

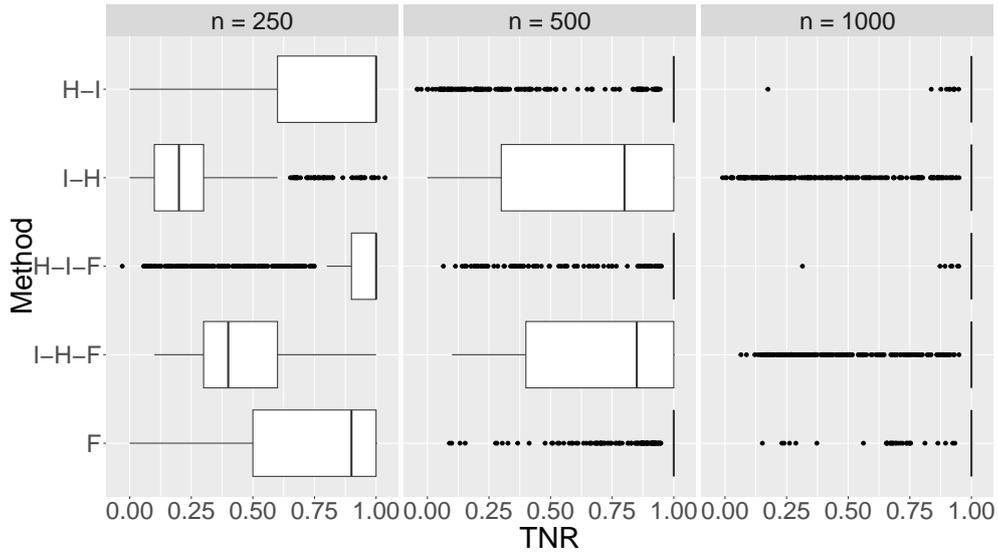


Figure 3: Simulation 1: boxplots for TNR (the true negative rate for the input variables) for each method by sample size.

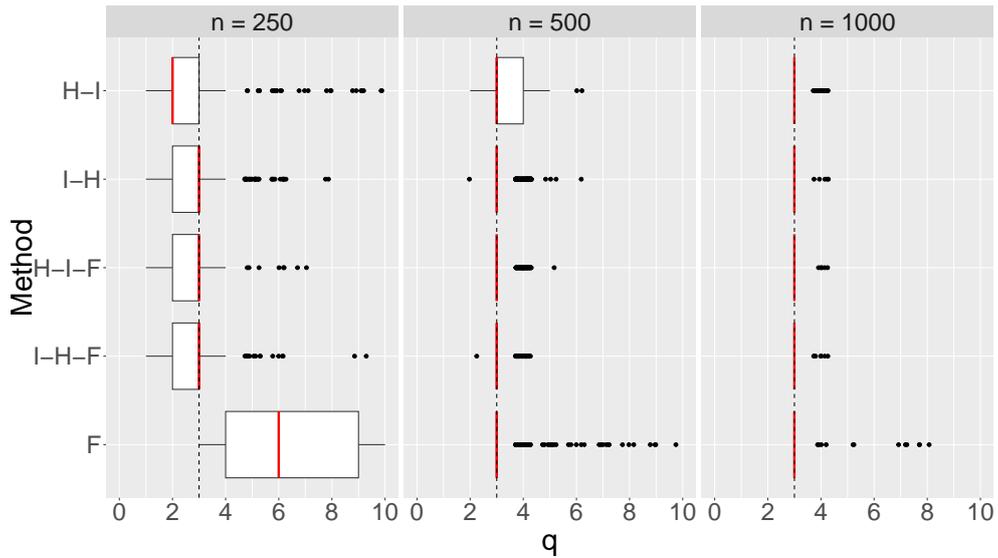


Figure 4: Simulation 1: boxplots for q (the number of hidden nodes selected) for each method by sample size. Median value highlighted in red. Dashed line indicates the true value of q .

networks of using out-of-sample performance. Three objective functions are investigated: BIC, AIC, and out-of-sample mean squared error (OOS). The AIC approach is the same as the proposed approach in Section 3.1, swapping BIC for $AIC = -2\ell(\hat{\theta}) + 2(K + 1)$. The OOS approach follows the same procedure, but with the objective function replaced by out-of-sample mean squared error, which is calculated on an additional validation data

Table 2: Simulation 2: model selection metrics.

n	Method	Input layer			Hidden layer		K (16)	OOS Test	PT
		TNR	FDR	PI	\bar{q} (3)	PH			
250	AIC	0.25	0.71	0.00	11.70	0.00	144	2.29	0.00
	BIC	0.87	0.15	0.72	2.66	0.54	16	0.86	0.43
	OOS	0.45	0.60	0.04	2.79	0.28	27	1.30	0.01
500	AIC	0.24	0.71	0.00	11.40	0.00	144	1.03	0.00
	BIC	0.96	0.05	0.90	3.05	0.95	16	0.53	0.85
	OOS	0.46	0.60	0.03	3.91	0.36	37	0.57	0.00
1000	AIC	0.27	0.70	0.00	11.40	0.00	141	0.76	0.00
	BIC	1.00	0.00	0.99	3.00	1.00	16	0.56	0.99
	OOS	0.53	0.57	0.02	3.72	0.46	34	0.57	0.00

Best values for a given sample size are highlighted in **bold**.

set that is 20% the size of the training data set, i.e., $\text{OOS} = \frac{1}{\tilde{n}} \sum_{i=1}^{\tilde{n}} (\tilde{y}_i - \text{NN}(\tilde{x}_i))^2$, where \tilde{n} is the number of observations in the validation data set with response variable \tilde{y}_i and covariate vector \tilde{x}_i . As before, $n_{\text{init}} = 5$ random initialisations are used. The results of the simulation study are shown in Table 2 and boxplots of TNR for the inputs and q for the different objective functions are given in Appendix G.

The results show that BIC far outperforms OOS and AIC in correctly identifying the correct FNN architecture. Using OOS as the model selection objective function almost never leads to correct neural network architecture being identified. This is due to the inability of the OOS to correctly identify and remove the unimportant covariates (TNR is always relatively low). Using AIC leads to even worse performance, and this is likely due to the weaker penalty on model complexity compared to BIC. It is also of interest to compare the approaches in terms of the size of the model selected and its out-of-sample performance. The median number of neural network parameters, K (note that the true value is $K = 16$), and the median out-of-sample mean squared error (OOS Test) evaluated on a test set are reported. The OOS Test is computed on an entirely new dataset (20% the size of the training set) that the OOS-optimising procedure was not exposed to. Interestingly, BIC-minimisation leads to the lowest OOS values on the test data. This is particularly noteworthy since this is achieved using approximately half as many parameters as the OOS-minimisation procedure. Boxplots highlighting the values of OOS Test and K are shown in Figures 5 and 6, respectively. Figure 5 also displays the OOS Test values for the true model (inputs x_1, x_2, x_3 and $q = 3$) and the full model (inputs x_1, x_2, \dots, x_{13} and $q = 10$); this allow us to evaluate the performance of selection compared to the full model, and how close we can get to the true model. The models selected using the BIC procedure have similar performance to the true model, particularly as the sample size increases. In contrast, the models selected using AIC have worse out-of-sample performance and significantly more parameters, and the performance is similar

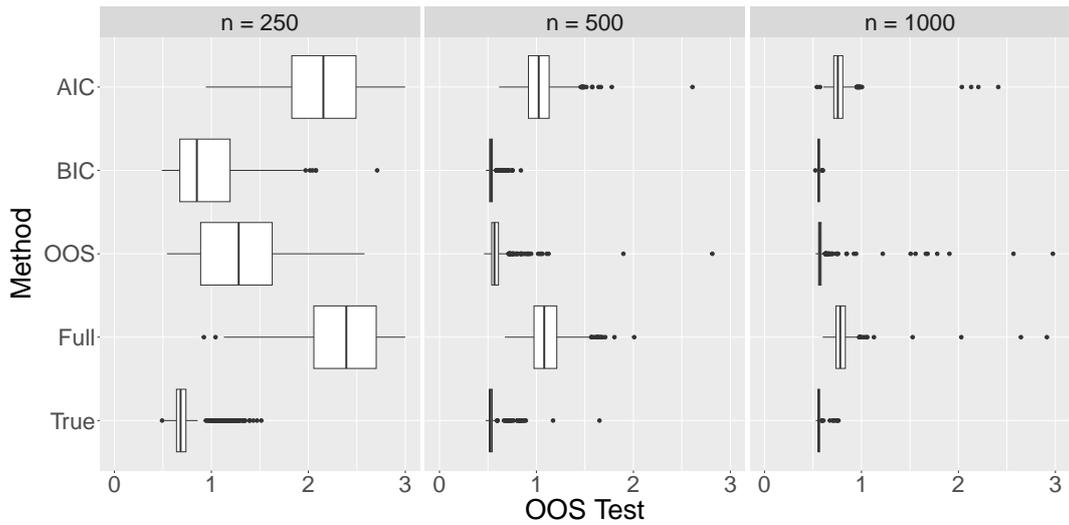


Figure 5: Simulation 2: boxplots for OOS Test for the models selected by each objective function; for comparison, the results for the true model (with inputs x_1, x_2, x_3 and $q = 3$) and the full model (with inputs x_1, x_2, \dots, x_{13} and $q = 10$).

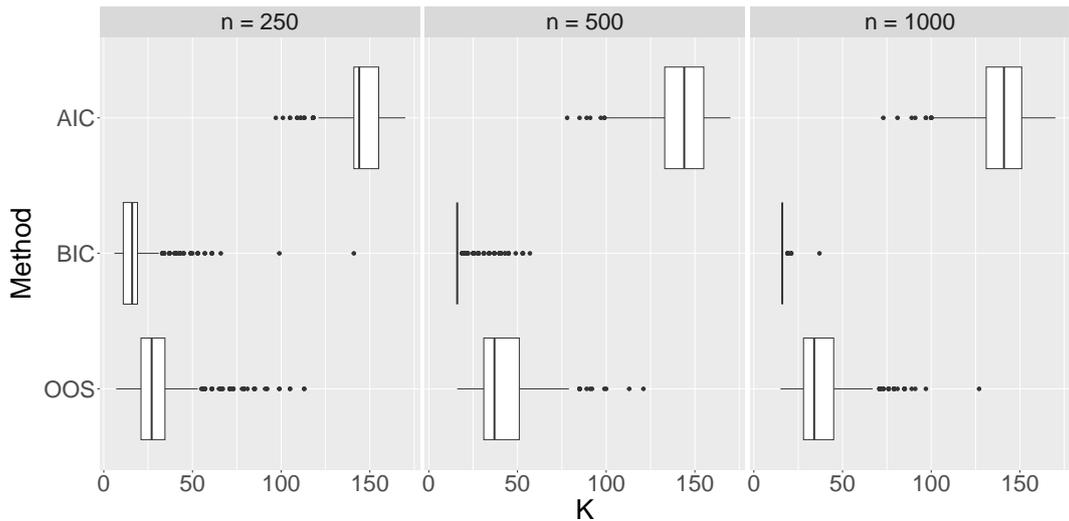


Figure 6: Simulation 2: boxplots for K (number of parameters) for the models selected by each objective function.

to fitting the full model.

We have also compared our proposed BIC-based selection procedure to two commonly used strategies for dealing with overfitting, namely, weight decay and early stopping. The results are deferred to Appendix H, where we have found that our proposed approach yields improved OOS Test values compared to these other two strategies.

4.3 Simulation 3: Data-generating Process is not a Neural Network

For this simulation study, we investigate the performance of the proposed H-I-F model selection procedure on a data set simulated from a data-generating process that is not a neural network:

$$y = x_1 - 0.75x_2^2 + 0.9x_3x_4x_5 + \varepsilon, \quad (4.3)$$

where $x_1, x_2, \dots, x_{10} \sim N(0, 1)$, i.e., there are five relevant and five irrelevant covariates, and $\sigma^2 = \text{Var}(\varepsilon) = 0.3$. For comparison, we have also performed stepwise model selection for a linear model using BIC. We applied this using the `stepAIC` function from the `MASS` R package with `k = log(n)` (Venables and Ripley, 2002). To compare with the H-I-F procedure, we also performed stepwise selection on a linear model with a search space containing (i) all terms up to three-way interactions (`step-lm-3`), (ii) all terms up to two-way interactions (`step-lm-2`), and (iii) only main effects (`step-lm-1`). Note that the first model is correctly specified, and the latter two are misspecified. For these linear models, we began the search with all possible terms in the model, and allowed the stepwise search to consider both the elimination of an included variable and the addition of an excluded variable at each step (i.e., `direction = "both"`). For the purpose of this study, when computing performance metrics (displayed in Table 3), we only considered whether or not relevant variables (x_1, \dots, x_5) and irrelevant variables (x_6, \dots, x_{10}) are selected. While the exact functional form of each selected variable is not considered, the OOS metrics facilitate model comparisons in the sense that lower OOS values imply a better approximation to the generating model (i.e., the functional form of input variables). In Table 3, as with earlier tables, the TNR, FDR, and PT selection metrics are shown, but, here, the TPR (true positive rate) metric is also shown. Moreover, we also show median number of parameters (K), the median out-of-sample mean squared error evaluated on a test set (OOS Test), and the median computational time (Time) for each approach.

From Table 3, we see that the proposed H-I-F procedure has a high true negative rate, a low false discovery rate, and the true positive rate increases with the sample size; consequently, the probability of selecting the true set of covariates (PT) increases with the sample size. At the highest sample size, the out-of-sample performance is very close to that of the correctly specified third order linear model (`step-lm-3`). Although this true `step-lm-3` model provides the lowest out-of-sample performance, its true negative and false discovery rates are relatively poor compared to the neural network, and, hence, the probability of selecting the true set of covariates does not approach one for the sample sizes we have considered. The selected `step-lm-3` model does have fewer parameters on average than the neural network model (at $n = 500$ and $n = 1000$), but the `step-lm-3` search is far more computationally intensive; this is due to the large number of possible interaction terms up to order three. It is important to note that the stepwise approaches for the linear models require the search space of models to be explicitly specified through the interaction and polynomial terms, and the performance of the misspecified (`step-lm-2` and `step-lm-1`) approaches is very poor. In contrast, the proposed H-I-F selection

Table 3: Simulation 3: Comparison of proposed model selection approach for neural networks with stepwise model selection for linear models for the data-generating process given by Equation 4.3.

Method	n	TPR	TNR	FDR	PT	K	OOS Test	Time (s)
H-I-F	250	0.53	0.90	0.13	0.02	11	2.12	12
H-I-F	500	0.78	0.95	0.04	0.50	43	0.52	22
H-I-F	1000	1.00	0.98	0.02	0.93	57	0.30	46
step-lm-3	250	1.00	0.13	0.45	0.04	61	0.73	53
step-lm-3	500	1.00	0.46	0.32	0.12	20	0.36	105
step-lm-3	1000	1.00	0.63	0.24	0.23	16	0.28	215
step-lm-2	250	0.85	0.35	0.43	0.00	15	1.95	2
step-lm-2	500	0.83	0.39	0.42	0.00	13	1.31	3
step-lm-2	1000	0.93	0.80	0.16	0.00	10	1.02	5
step-lm-1	250	0.20	1.00	0.00	0.00	2	3.55	0
step-lm-1	500	0.20	1.00	0.00	0.00	2	2.46	0
step-lm-1	1000	0.40	1.00	0.00	0.00	3	1.98	0

Time (s), median time to completion in seconds (carried out on an Intel[®] Core[™] i5-10210U Processor).

approach does not require these terms to be explicitly specified, but still achieves very good out-of-sample performance since complex functional relationships and interactions are captured in a more automatic manner within the neural network structure.

5 Application to Data

Airbnb is an online marketplace that provides both short-term and long-term rentals. Data relating to the rental listings can be obtained from Inside Airbnb (<http://insideairbnb.com>). Here, we focus on rental listings in the Dún Laoghaire–Rathdown area of Dublin on the seventh of September 2023, and aim to implement our proposed model selection approach, and determine factors that may be associated with the listing’s price. The data consists of information relating to 625 rental listings, and the following explanatory variables: the number of people the rental accommodates (`accommodates`), the rental’s review rating (`rating`), the number of reviews per month (`num_reviews`), an indicator of whether the rental is an entire home or a private room (`room_type`; 0 for an entire home, 1 for a private room), an indicator of whether or not the host is a “superhost” (`superhost`; i.e., top-performing Airbnb hosts, where performance is based on reviews, responsiveness and their cancellation rate), the total number of Airbnb listings that the host has (`num_listings`), an indicator of whether or not the listing is instantly bookable (`instant`), and the latitude (`latitude`) and longitude (`longitude`) of the rental. The

Table 4: Dublin Airbnb: selected versus full model comparison.

	p	q	K	BIC	OOS
Selected	3	2	11	884.2	0.25
Full	9	10	111	1136.3	0.48

response variable is the natural logarithm of the price per night of each rental (`lnprice`). The data is available in our R package `selectnn` (McInerney and Burke, 2022).

The dataset has been randomly split into a training set and test set with a 80%–20% split, respectively, and all continuous variables have been standardised (based on the training data) to have zero mean and unit variance. The model selection procedure was implemented with $n_{\text{init}} = 10$ and $q_{\text{max}} = 10$. For comparison purposes, the model found by our proposed model selection procedure is compared to fitting an FNN with all inputs and the maximum number of hidden nodes considered. For both models (selected and full), we report the number of input nodes (p), the number of hidden nodes (q), the total number of parameters (K), the BIC, and the out-of-sample mean squared error (OOS) computed using the test set. For the covariates that are selected, we also report: (i) relative covariate importance via the change in BIC (ΔBIC) upon removal of that covariate, and (ii) a simple covariate effect ($\hat{\tau}$) as measured by the change in the average predicted response going from lower to higher covariate values (below/above median for numeric covariates and 0/1 for binary covariates). See Appendix I for more detail on these measures.

Our proposed procedure selects two hidden nodes and includes three covariates: `accommodates`, `num_reviews` and `room_type`. As shown in Table 4, the selected model has 100 fewer parameters than the full model, while also having a much lower BIC value and a lower out-of-sample mean squared error. The BIC differences and covariate effects (and their associated bootstrapped confidence intervals) for the variables that remain in the model are reported in Table 5. Using ΔBIC as a measure of variable importance, we find that `accommodates` is the most important variable with $\Delta\text{BIC}_{\text{rm}} = 200.16$. Based on its effect ($\hat{\tau}_{\text{accommodates}} = 1.38$), the more people the listing accommodates, the higher the price per night. The binary variable `room_type` has a negative effect, which suggests that the listing price of a private room is lower than an entire house, on average. The other covariate, `num_reviews`, is more important than `room_type` as judged by its ΔBIC value, but the confidence interval for its covariate effect includes zero. This suggests that `num_reviews` has a non-linear effect that cannot be seen in an overall average change in the predicted response.

The selected model dropped six covariates (from a set of nine possible covariates). While the underlying selection procedure cannot guarantee that this model minimises the BIC, and an exhaustive search through all sub-models is computationally expensive, we have nevertheless carried out such a search for the purpose of comparison. To this end, we fitted the model with all covariates included, all nine models that arise by dropping one of

Table 5: Dublin Airbnb: covariate effects and BIC differences.

	$\hat{\tau}$ (95% CI)	ΔBIC
<code>accommodates</code>	1.38 (1.30, 1.47)	200.16
<code>num_reviews</code>	0.07 (-0.07, 0.21)	89.56
<code>room_type</code>	-1.44 (-1.51, -1.37)	64.02

each of the covariates, all 36 models that arise when pairs of covariates are dropped, all 84 models that arise when triples of covariates are dropped, and so on, for each hidden layer size, $q = 1, \dots, 10$. Each model was allowed $n_{\text{init}} = 10$ random initialisations, mirroring that of our selection procedure.

Figure 7 shows the BIC for each model where each point corresponds to a different input-layer hidden-layer combination; for comparison, the model selected by our procedure is indicated using a box. First, note that there is a subset of models with relatively large BIC values. Each of these models are missing the variables `accommodates` and `room_type`, which further highlights their importance in the model. It is clear that the proposed selection procedure has indeed found a model with a BIC value that is among the lowest of the alternative models we have considered. That being said, the exhaustive search did return two models with lower BIC values ($\Delta\text{BIC} = -4.4$ and $\Delta\text{BIC} = -2.7$). These models are more complex than the model selected by our procedure (with $q = 2$ and $p = 3$) as they have $q = 3$ and $q = 2$ hidden nodes with $p = 4$ and $p = 5$ input nodes, respectively. We note that the out-of-sample predictive performance is very similar across all three of these models.

6 Discussion

FNNs have become very popular in recent years and have the potential to capture more complex covariate effects than traditional statistical models. However, model selection procedures are of the utmost importance in the context of FNNs since their flexibility may increase the chance of over-fitting; indeed, the principle of parsimony is very common throughout statistical modelling more generally. Therefore, we have proposed a statistically-motivated neural network selection procedure by assuming an underlying (normal) error distribution, which then permits BIC minimisation. More specifically, our procedure involves a hidden-node selection phase, followed by an input-node (covariate) selection phase, followed by a final fine-tuning phase. We have made this procedure available in our `selectnn` package in R (McInerney and Burke, 2022).

Through extensive simulation studies, we have found that that (i) the order of selection (input versus hidden layer) is important, with respect to the probability of recovering the true model and the computational efficiency, (ii) the addition of a fine-tuning stage provides a non-negligible improvement while not significantly increasing the computational burden, (iii) using the BIC is necessary to asymptotically converge to the true model,

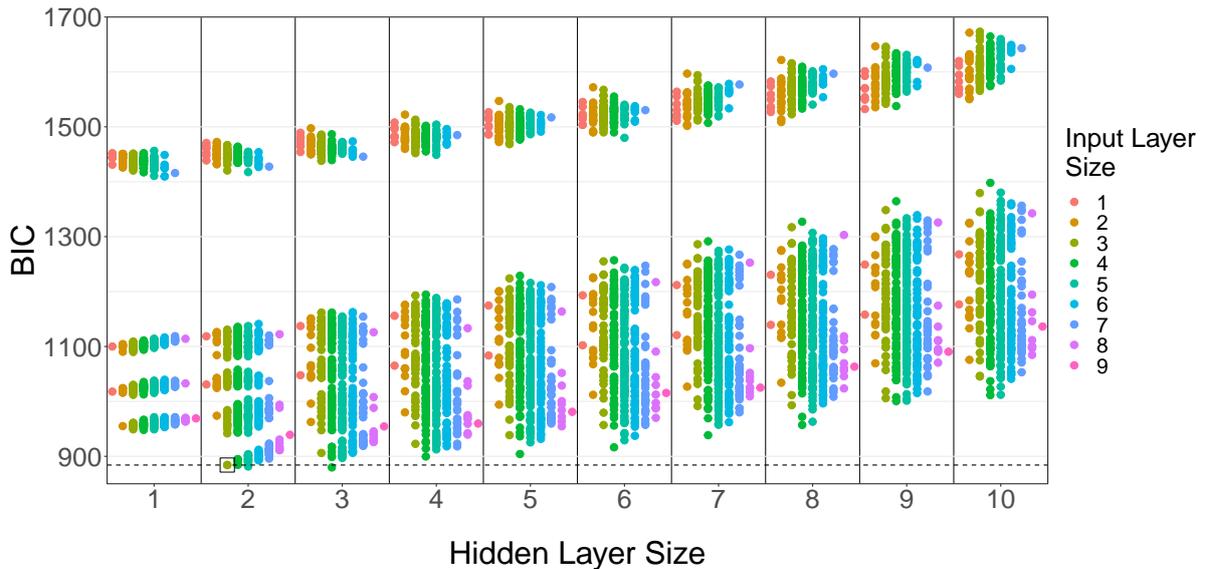


Figure 7: Dublin Airbnb: BIC of models for different input-layer and hidden-layer combinations. Points are coloured according to the input layer size. The model selected by our procedure is enclosed in a box and the horizontal dashed line indicates the BIC for this model.

and (iv) although the models selected using BIC have fewer parameters than those selected using out-of-sample performance, they have comparable, and sometimes improved, predictivity. We suggest that statistically-orientated model selection approaches are necessary in the application of neural networks — just as they are in the application of more traditional statistical models — and we have demonstrated the favourable performance of our proposal.

In its current form, a limitation of the proposed procedure is that, due to its stepwise nature, it would be more computationally intensive when dealing with larger models and datasets. We expect that randomisation and/or divide-and-conquer throughout the selection phases would be required in more complex problems involving many covariates and/or hidden layers, and adaptations may also be required for stochastic optimisation procedures used on much larger datasets. Nevertheless, neural networks are still valuable in more traditional (smaller) statistical problems for which procedures such as ours will lead to more insightful outputs. Furthermore, the implementation of statistical approaches more broadly (such as uncertainty quantification and hypothesis testing) in neural network modelling will be crucial for the enhancement of these insights. This will be the direction of our future work.

Acknowledgement

This publication has emanated from research conducted with the financial support of Science Foundation Ireland under Grant number 18/CRT/6049. The second author was supported by the Confirm Smart Manufacturing Centre (<https://confirm.ie/>) funded by Science Foundation Ireland (Grant Number: 16/RC/3918). For the purpose of Open Access, the author has applied a CC BY public copyright licence to any Author Accepted Manuscript version arising from this submission.

References

- Abiodun, O. I., Jantan, A., Omolara, A. E., Dada, K. V., Mohamed, N. A., and Arshad, H. (2018). State-of-the-art in artificial neural network applications: A survey. *Heliyon*, 4(11):e00938.
- Akaike, H. (1998). *Information Theory and an Extension of the Maximum Likelihood Principle*, pages 199–213. Springer New York, New York, NY.
- Anderson, D. and Burnham, K. (2004). Model selection and multi-model inference. *Second*. NY: Springer-Verlag.
- Bishop, C. M. et al. (1995). *Neural networks for pattern recognition*, chapter 9, pages 353–354. Oxford university press.
- Breiman, L. (2001). Statistical Modeling: The Two Cultures (with comments and a rejoinder by the author). *Statistical Science*, 16(3):199 – 231.
- Chandrashekar, G. and Sahin, F. (2014). A survey on feature selection methods. *Computers & Electrical Engineering*, 40(1):16–28.
- Cheng, B. and Titterton, D. M. (1994). Neural networks: A review from a statistical perspective. *Statistical Science*, 9(1):2–30.
- Cybenko, G. (1989). Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals and Systems*, 2(4):303–314.
- Efron, B. (2020). Prediction, estimation, and attribution. *International Statistical Review*, 88(S1):S28–S59.
- Elder, J. F. (2003). The generalization paradox of ensembles. *Journal of Computational and Graphical Statistics*, 12(4):853–864.
- Fan, J. and Lv, J. (2010). A selective overview of variable selection in high dimensional feature space. *Statistica Sinica*, 20(1):101–148.

- Fisher, R. A. and Russell, E. J. (1922). On the mathematical foundations of theoretical statistics. *Philosophical Transactions of the Royal Society of London. Series A, Containing Papers of a Mathematical or Physical Character*, 222(594-604):309–368.
- Goldberg, Y. (2016). A primer on neural network models for natural language processing. *Journal of Artificial Intelligence Research*, 57:345–420.
- Heinze, G., Wallisch, C., and Dunkler, D. (2018). Variable selection – a review and recommendations for the practicing statistician. *Biometrical Journal*, 60(3):431–449.
- Hooker, G. and Mentch, L. (2021). Bridging breiman’s brook: From algorithmic modeling to statistical learning. *Observational Studies*, 7(1):107–125.
- Hornik, K., Stinchcombe, M., and White, H. (1989). Multilayer feedforward networks are universal approximators. *Neural Networks*, 2(5):359–366.
- Koenen, N. and Wright, M. N. (2024). Interpreting deep neural networks with the package insight. *arXiv preprint arXiv:2306.10822*.
- LeCun, Y., Bengio, Y., and Hinton, G. (2015). Deep learning. *Nature*, 521(7553):436–444.
- McInerney, A. and Burke, K. (2022). *selectnn: A Statistically-Based Approach to Neural Network Model Selection*. R package version 0.0.0.9000.
- Miller, A. (2002). *Subset selection in regression*. chapman and hall/CRC.
- Murata, N., Yoshizawa, S., and Amari, S. (1994). Network information criterion-determining the number of hidden units for an artificial neural network model. *IEEE Transactions on Neural Networks*, 5(6):865–872.
- Pang, G., Shen, C., Cao, L., and Hengel, A. V. D. (2021). Deep learning for anomaly detection: A review. *ACM Computing Surveys (CSUR)*, 54(2):1–38.
- Pontes, F., Amorim, G., Balestrassi, P., Paiva, A., and Ferreira, J. (2016). Design of experiments and focused grid search for neural network parameter optimization. *Neurocomputing*, 186:22–34.
- Ripley, B. and Venables, W. (2022). nnet: Feed-forward neural networks and multinomial log-linear models. *R package version*, 7.3-17.
- Ripley, B. D. (1993). Statistical aspects of neural networks. In Nielsen, B. O. E., Jensen, J. L., and Kendall, W. S., editors, *Networks and Chaos: Statistical and Probabilistic Aspects*, pages 40–123. Chapman & Hall.
- Ripley, B. D. (1994). Neural networks and related methods for classification. *Journal of the Royal Statistical Society: Series B (Methodological)*, 56(3):409–437.

- Rügamer, D., Kolb, C., and Klein, N. (2020). Semi-structured deep distributional regression: Combining structured additive models and deep learning. *arXiv preprint arXiv:2002.05777*.
- Schwarz, G. (1978). Estimating the dimension of a model. *The Annals of Statistics*, 6(2):461–464.
- Sun, Y., Song, Q., and Liang, F. (2022). Learning sparse deep neural networks with a spike-and-slab prior. *Statistics & Probability Letters*, 180:109246.
- Tibshirani, R. (1996). Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society: Series B (Methodological)*, 58(1):267–288.
- Tran, M.-N., Nguyen, N., Nott, D., and Kohn, R. (2020). Bayesian deep net glm and glmm. *Journal of Computational and Graphical Statistics*, 29(1):97–113.
- Venables, W. N. and Ripley, B. D. (2002). *Modern Applied Statistics with S*. Springer, New York, fourth edition. ISBN 0-387-95457-0.
- Voulodimos, A., Doulamis, N., Doulamis, A., and Protopapadakis, E. (2018). Deep learning for computer vision: A brief review. *Computational Intelligence and Neuroscience*, 2018.
- White, H. (1989). Learning in Artificial Neural Networks: A Statistical Perspective. *Neural Computation*, 1(4):425–464.
- Ye, J. (1998). On measuring and correcting the effects of data mining and model selection. *Journal of the American Statistical Association*, 93(441):120–131.

A Neural Network Degrees of Freedom

The use of the BIC for model selection introduces the question of degrees of freedom for neural networks. In our procedure, we define the degrees of freedom to be the number of parameters in the model, K . From our simulation results in Section 4 of the main paper, we see that this leads to consistent model selection. However, other approaches to defining the degrees of freedom for neural networks exist. Ye (1998) defined the concept of generalised degrees of freedom (GDF) as

$$\text{GDF} = \frac{\sum_{i=1}^n \text{cov}(y_i, \hat{y}_i)}{\sigma^2},$$

where \hat{y}_i are the predicted values from the model; note that the computation of GDF in practice is based refitting the model many times to datasets with slightly perturbed values of y_i (Elder, 2003; Ye, 1998). Murata et al. (1994) introduced a network information criterion (NIC) whose penalty for model complexity is given by

$$\text{EDF} = \text{tr}(GQ^{-1}),$$

where $Q = E[\nabla_{\theta}\nabla_{\theta}^T\ell(\theta)]$, $G = \text{Var}[\nabla_{\theta}\ell(\theta)]$ and $\text{tr}(\cdot)$ denotes the trace operator. In this Appendix, we consider the behaviour of these degrees of freedom formulae using a variety of simulations. The first simulation study investigates the values of EDF and GDF for different neural network architectures when the model is correctly specified. The second simulation study also investigates EDF and GDF, but for neural networks that are incorrectly specified. In particular, we focus on cases when the neural network fit to the data is larger than the true data-generating model. Finally, we implement both methods as the degrees-of-freedom term within our model selection procedure, and compare the results to our proposed use of the number of parameters as the degrees of freedom (i.e., the classical BIC penalty).

For the first simulation study, the degrees of freedom are estimated for various neural network architectures. Both the size of the input layer and the hidden layer are varied with $p, q \in \{2, 4, 6, 8\}$. For each architecture, a correctly specified neural network model is fit to the data. Sample size is varied with $n \in \{250, 500, 1000, 2000\}$ and 100 simulation replicates are carried out. The results for each architecture are displayed in Figure 8.

It is clear that the GDF, EDF, and K (the number of parameters) closely align with each other in the scenarios we have considered. In the more complicated (larger) models, there is more variability in the GDF and EDF values at smaller samples, but they converge to K with the sample size. Interestingly, the convergence to K is from below, implying that, if GDF or EDF are used within a BIC selection procedure, they will penalise complexity less than when using K (and, hence, select more complex models than those selected using the classical BIC).

Aside from the reduced penalisation relative to K , it is also worth noting that GDF and EDF have other drawbacks. First, GDF is quite computationally intensive to compute; a plot of the average time taken to compute GDF is given in Figure 9. The computational

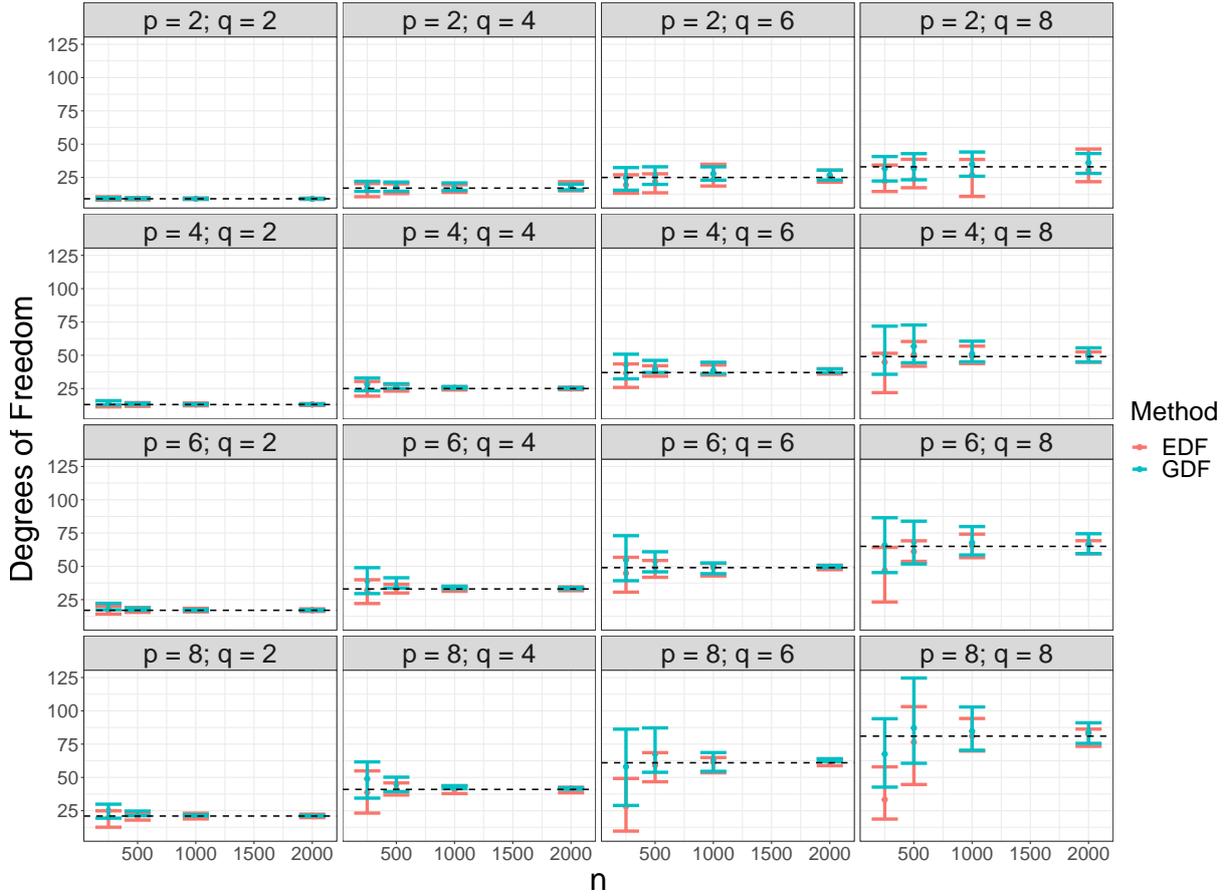


Figure 8: The average EDF and GDF values and their associated 2.5 and 97.5 quantiles versus sample size for different neural network architectures. The horizontal dashed line represents the number of parameters for each architecture.

times observed (which further increase with sample size and model complexity) render the use of GDF within model selection less feasible since a degrees-of-freedom value is needed at each step of the selection procedure. As for the EDF computation, this requires the inversion of the Hessian matrix of the neural network parameters, which is not possible when there are redundancies present in the model; this is something that becomes more likely in more complex models. Figure 10 displays the proportion of replicates where the EDF could not be computed for each architecture and sample size. It is clear that larger sample sizes are required for more complicated architectures in order to ensure stable computation of EDF.

The second simulation study is similar to the first, but now the neural network architecture is misspecified. The true data-generating model has $p = 4$ input nodes and $q = 4$ hidden nodes. Neural networks of various architectures are fit to the data, with $p, q \in \{2, 4, 6, 8\}$. The results are displayed in Figure 11. For all models where the hidden layer is correctly specified ($q = 4$), the results are similar to Figure 8, i.e., the GDF and

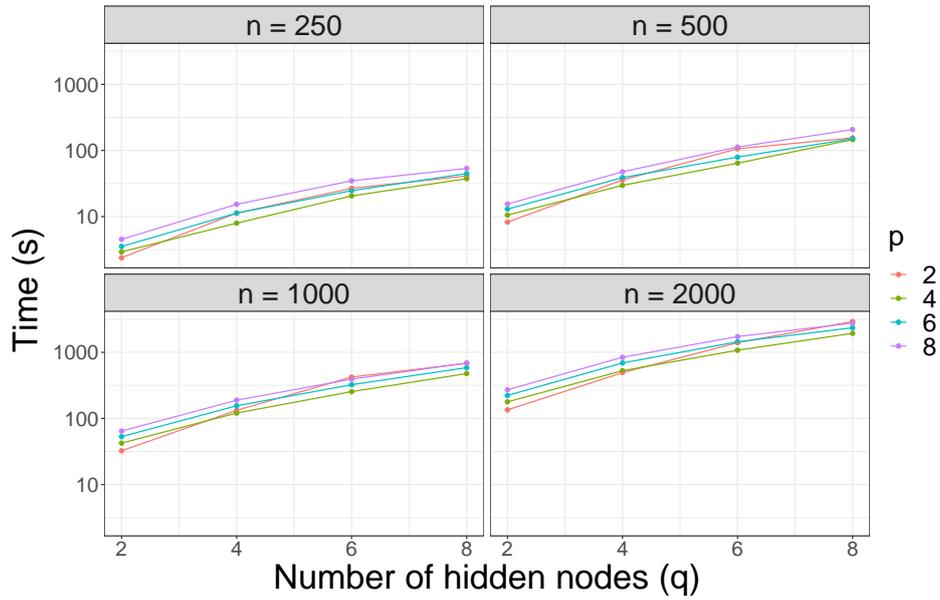


Figure 9: The average time taken to compute GDF for different architectures and sample sizes.

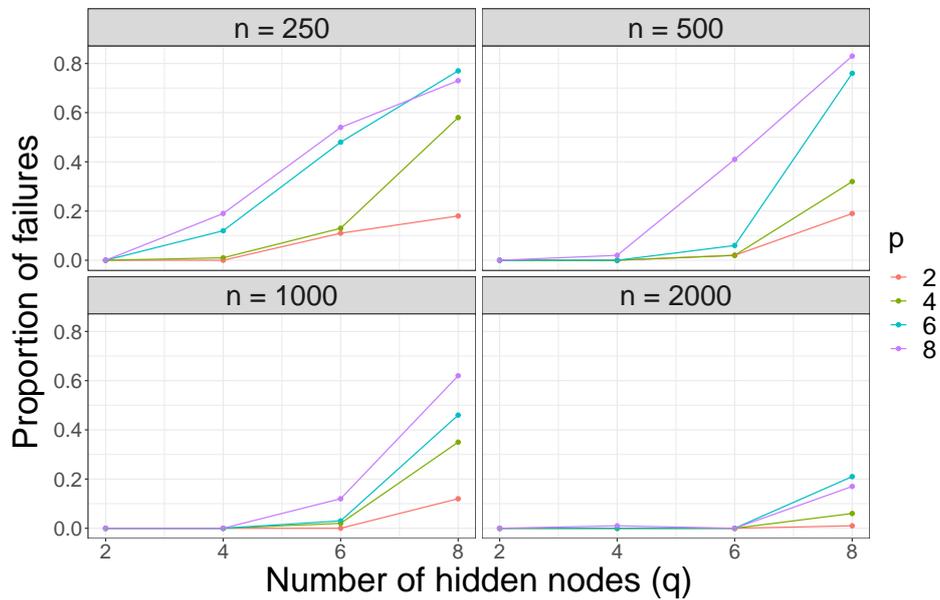


Figure 10: The proportion of replicates that the computation of EDF failed due to a non-invertible Hessian matrix.

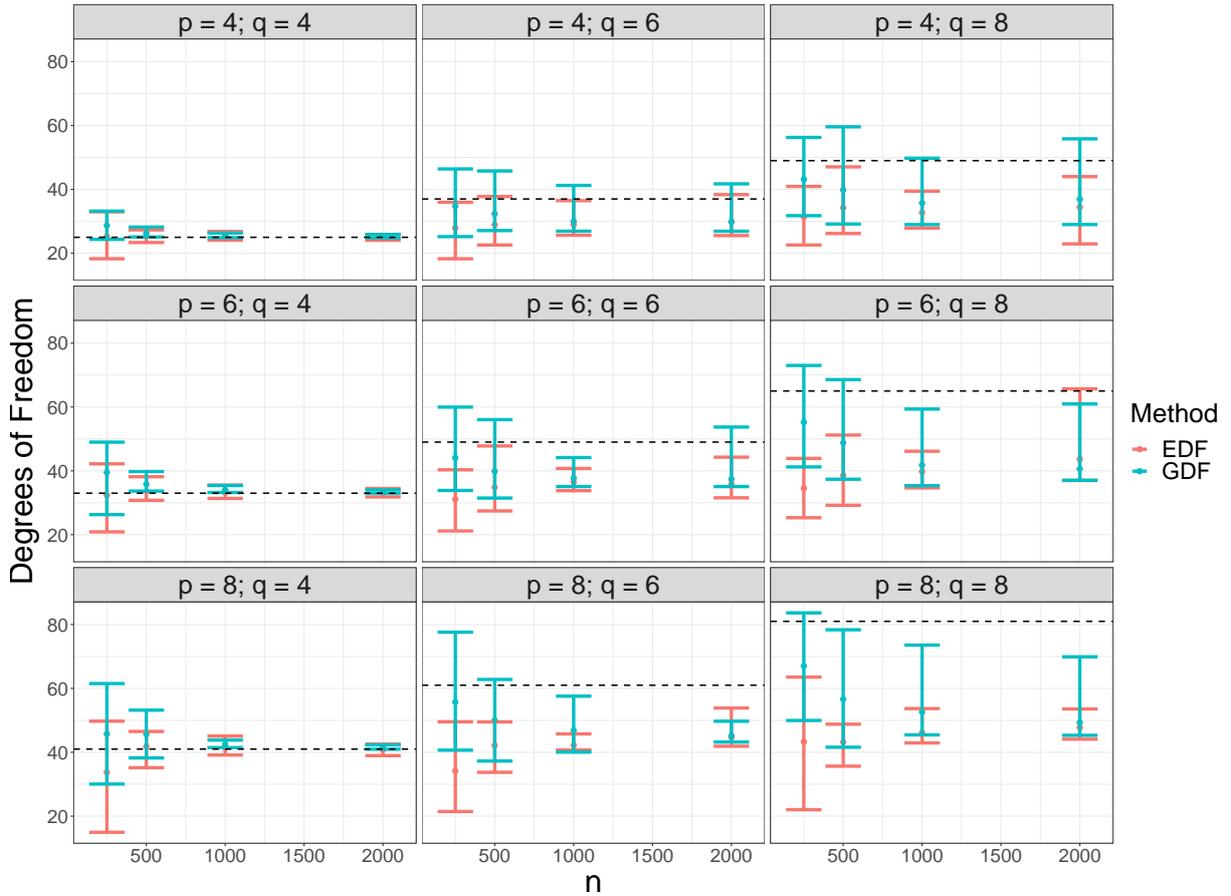


Figure 11: The average EDF and GDF values and their associated 2.5 and 97.5 quantiles versus sample size for different neural network architectures. The horizontal dashed line represents the number of parameters for each architecture.

EDF approaches align with the number of parameters, K . However, when the number of hidden nodes is incorrectly specified, EDF and GDF do not tend to the number of parameters. In these scenarios, the degrees of freedom is lower than the number of parameters in the model; this suggests redundancies in the fitted model.

Finally, we compared all three approaches to defining the degrees of freedom within our proposed model selection procedure. Due to the computational expense of GDF, a simpler simulation set up is used compared to the simulation study in the main paper. Here, the true data-generating model is the same as the neural network used in Sections 4.1 and 4.2. However, only two unimportant inputs and a $q_{\max} = 5$ are considered. The results of the model selection simulation are displayed in Table 6.

From the simulation results, it is clear that using GDF or EDF within the BIC penalty leads to reduced performance in the model selection procedure compared with using K . This is likely due to the penalty being weaker when there is redundancy present (as seen in Figure 11), which leads to the selection of larger-than-required models (evidenced in

the inflated false discovery rates in both the input and hidden nodes). The median out-of-sample mean squared error (OOS) evaluated on a test set (20% the size of the training set) is also reported. The models selected using K as the degrees-of-freedom term have better predictive performance than the models selected using the EDF and GDF approaches (albeit the OOS values appear to converge with the sample size).

Table 6: Simulation results: model selection metrics for different degrees-of-freedom approaches.

n	Method	Input layer			Hidden layer		OOS	PT
		TNR	FDR	PI	$\bar{q}(3)$	PH		
250	K	0.85	0.07	0.77	3.24	0.47	0.89	0.38
	GDF	0.56	0.20	0.29	4.93	0.05	1.10	0.00
	EDF	0.22	0.33	0.04	5.80	0.02	1.49	0.00
500	K	0.94	0.03	0.90	3.08	0.92	0.53	0.82
	GDF	0.74	0.12	0.55	4.66	0.15	0.60	0.10
	EDF	0.60	0.18	0.40	5.30	0.03	0.62	0.02
1000	K	0.97	0.01	0.95	3.04	0.98	0.47	0.98
	GDF	0.74	0.12	0.57	5.61	0.02	0.51	0.02
	EDF	0.74	0.12	0.59	4.78	0.02	0.50	0.01

B Simulation Results with Correlated Data

The performance of the proposed H-I-F approach is further investigated in the setting where there is correlation among the covariates. Here, the data-generating process is the same as in the main paper, however, the data is generated such that the covariates are multivariate normal, i.e., $x_1, x_2, \dots, x_{13} \sim \text{MVN}$ wherein $\text{corr}(x_j, x_k) = 0.7^{|j-k|}$. The non-zero covariates are x_1, x_7, x_{13} . The results corresponding to Simulation 1 and 2 are shown in Tables 7 and 8, respectively.

Table 7: Simulation results: model selection metrics with correlated data for different model selection approaches.

n	Method	Input layer			Hidden layer		
		TNR	FDR	PI	\bar{q} (3)	PH	PT
250	H-I	0.82	0.38	0.63	2.89	0.52	0.38
	I-H	0.27	0.71	0.03	2.80	0.57	0.02
	H-I-F	0.84	0.35	0.60	3.07	0.82	0.53
	I-H-F	0.35	0.71	0.02	2.77	0.54	0.02
	F	0.68	0.52	0.35	7.52	0.24	0.18
500	H-I	0.96	0.13	0.90	3.23	0.76	0.73
	I-H	0.65	0.54	0.44	2.99	0.95	0.43
	H-I-F	0.97	0.10	0.91	3.06	0.94	0.86
	I-H-F	0.68	0.53	0.42	2.99	0.92	0.40
	F	0.97	0.10	0.88	3.15	0.92	0.83
1000	H-I	1.00	0.00	0.99	3.00	1.00	0.98
	I-H	0.86	0.32	0.76	2.99	0.99	0.76
	H-I-F	1.00	0.00	0.99	3.00	1.00	0.99
	I-H-F	0.88	0.29	0.78	3.00	0.98	0.77
	F	0.99	0.03	0.97	3.03	0.98	0.96

Table 8: Simulation results: model selection metrics for the proposed approach (H-I-F) with correlated data for different objective functions.

n	Method	Input layer			Hidden layer			OOS Test	PT
		TNR	FDR	PI	\bar{q} (3)	PH	K (16)		
250	AIC	0.12	0.75	0.00	12.17	0.00	157	2.17	0.00
	BIC	0.84	0.35	0.60	3.07	0.82	16	0.57	0.53
	OOS	0.41	0.66	0.02	4.12	0.35	39	0.87	0.00
500	AIC	0.13	0.74	0.00	11.71	0.00	155	1.25	0.00
	BIC	0.97	0.10	0.91	3.06	0.94	16	0.61	0.86
	OOS	0.49	0.63	0.02	3.74	0.44	37	0.64	0.00
1000	AIC	0.14	0.74	0.00	11.77	0.00	155	0.83	0.00
	BIC	1.00	0.00	0.99	3.00	1.00	16	0.51	0.99
	OOS	0.52	0.62	0.02	3.78	0.43	34	0.52	0.00

C Simulation: Number of Initialisations

Since FNNs have a complex optimisation surface (the log-likelihood function), each model fit is supplied with n_{init} random initial vectors with the aim of avoiding local maxima. Of course, larger values of n_{init} improve the chances of finding the global maximum but increase the computational expense. In the previous simulations, we fixed $n_{\text{init}} = 5$, whereas, here, we vary it at $n_{\text{init}} \in \{1, 5, 10\}$ using the proposed H-I-F BIC-minimisation procedure. Plots of the probability of choosing the correct number of hidden nodes (PH), the probability of choosing the correct set of inputs (PI), and the probability of choosing the overall true model (PT) for different values of n and n_{init} are shown in Figure 12. Also, Figure 13 displays boxplots of the computational time for each scenario. The corresponding table of simulation results is given in Table 9.

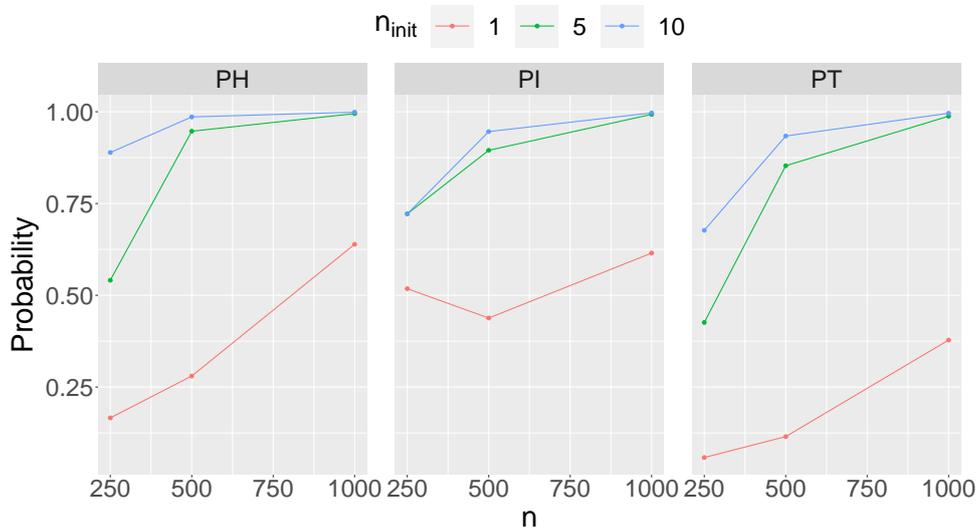


Figure 12: Simulation for number of initialisations: line plots for PH (the probability of choosing the correct number of hidden nodes), PI (the probability of choosing the correct set of inputs) and PT (the probability of choosing the overall true model) for different values of n and n_{init} .

From the plots, we can clearly see the trade-off between better model selection and worse computational efficiency as n_{init} increases. We would certainly recommend $n_{\text{init}} > 1$ initial vectors since the results are poor for $n_{\text{init}} = 1$. Beyond this, the choice might be based on the computational constraints in a given practical setting, but we note, in particular, that larger values of n_{init} are more important in smaller sample sizes.

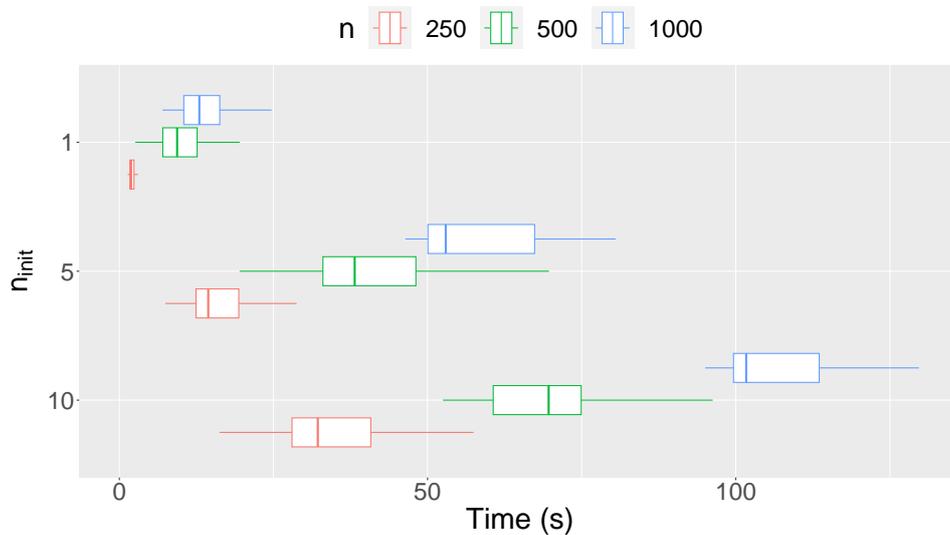


Figure 13: Simulation for number of initialisations: boxplots of computational time (s) for different values of n and n_{init} .

Table 9: Simulation for number of initialisations: model selection metrics.

n	n_{init}	Time (s)	Input layer			Hidden layer		
			TNR	FDR	PI	\bar{q} (3)	PH	PT
250	1	1.90	0.80	0.23	0.52	2.20	0.17	0.06
	5	14.42	0.87	0.15	0.72	2.66	0.54	0.43
	10	32.30	0.89	0.14	0.72	3.05	0.89	0.68
500	1	9.37	0.73	0.31	0.44	3.82	0.28	0.12
	5	38.18	0.96	0.05	0.90	3.05	0.95	0.85
	10	70.69	0.98	0.03	0.95	3.01	0.99	0.93
1000	1	12.98	0.89	0.16	0.62	3.40	0.64	0.38
	5	54.39	1.00	0.00	0.99	3.00	1.00	0.99
	10	117.32	1.00	0.00	1.00	3.00	1.00	1.00

Time (s), median time to completion in seconds (carried out on an Intel[®] Core[™] i5-10210U Processor). Best values for a given sample size are highlighted in **bold**.

D Relationship Between the Structure of the Input and Hidden Layer

In Simulation 1 (Section 4.1), there appears to be a relationship between the structure of one layer on the probability of selecting the correct structure for the other layer. For example, when $n = 500$, performing input-layer selection after first performing hidden-layer selection (H-I) results in a much higher probability of selecting the correct set of input nodes (PI = 0.83) in comparison to performing input-layer selection first (I-H) (PI = 0.43). This suggests that input-layer selection is improved when the hidden layer is closer to its correct structure. In order to investigate this further, we performed two simulation studies: one to explore the relationship between the input-layer structure and the probability of selecting the correct number of hidden nodes, and another to explore the relationship between the hidden-layer structure and the probability of selecting the correct set of input nodes. In each simulation study, the response is generated from an FNN with known “true” architecture containing five important inputs, x_1, x_2, \dots, x_5 , and $q = 5$ hidden nodes. Within each simulation, every scenario is implemented for 1,000 replicates, using a sample size of $n = 500$.

The aim of the first simulation study is to investigate the effect of adding additional unimportant covariates to the set of important covariates on hidden-node selection. All important covariates remain in the data, while the number of unimportant inputs, n_{unimp} , is varied from zero up to ten. The hidden-node selection step (Algorithm 2) is implemented for each replicate, with a maximum of 10 hidden nodes being considered. The probability of choosing the correct number of hidden nodes (PH) is then calculated. The results are displayed in Table 10 and Figure 14. It is clear that the more unimportant covariates that are included in the data, the lower the probability in recovering the correct hidden-layer structure.

Table 10: Probability of selecting the correct hidden-layer structure.

n_{unimp}	0	1	2	3	4	5	6	7	8	9	10
PH	0.86	0.78	0.74	0.66	0.54	0.40	0.43	0.40	0.39	0.27	0.23

n_{unimp} , the number of unimportant covariates; PH, the probability of selecting the correct number of hidden nodes.

The second simulation study aims to investigate the effect of the number of hidden nodes on input-node selection. Ten additional unimportant inputs, x_6, x_7, \dots, x_{15} , are added to the data. The number of hidden nodes is varied from one up to ten. The input-node selection step (Algorithm 3) is implemented for each replicate, and the probability of choosing the correct set of input nodes (PI) is calculated. The results are displayed in Table 11 and Figure 15. We find that the closer the hidden layer is to having the correct number of hidden nodes, the greater the probability of recovering the correct set of input nodes. Both simulation studies verify that model selection for one layer is dependent on

the structure of the other layer, and, hence, justifies the use of a fine-tuning phase after performing input- and hidden-node selection.

Table 11: Probability of selecting the correct input-layer structure.

q	1	2	3	4	5	6	7	8	9	10
PI	0.01	0.53	0.59	0.58	0.71	0.60	0.63	0.66	0.63	0.58

q , the number of hidden nodes; PI, the probability of selecting the correct set of input nodes.

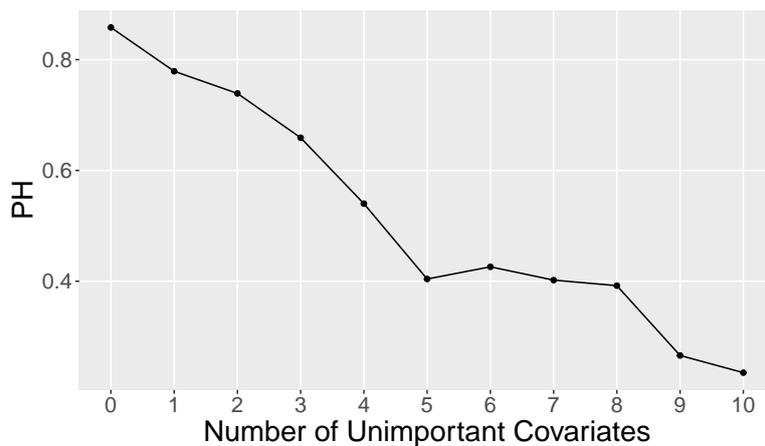


Figure 14: Number of unimportant covariates versus the probability of selecting the correct hidden-layer structure.

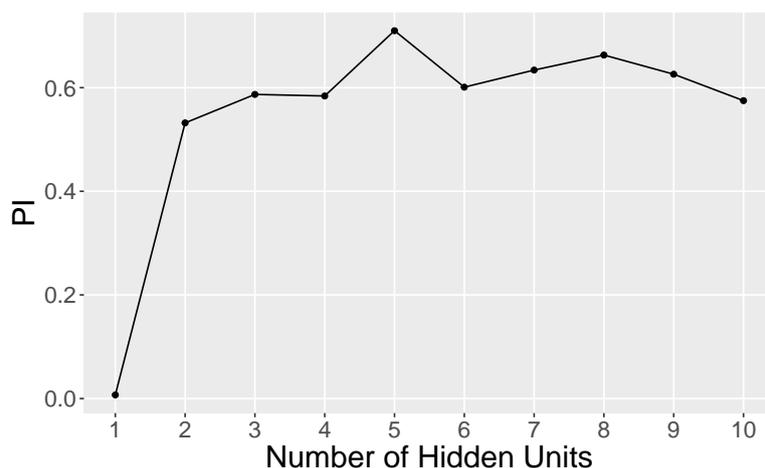


Figure 15: Number of hidden nodes versus the probability of selecting the correct input-layer structure.

E Simulation 1: Boxplots of Computational Time for Each Model Selection Method

This section contains the boxplots associated with the computational time for the different model selection approaches, and corresponds to Table 1 in the main paper.

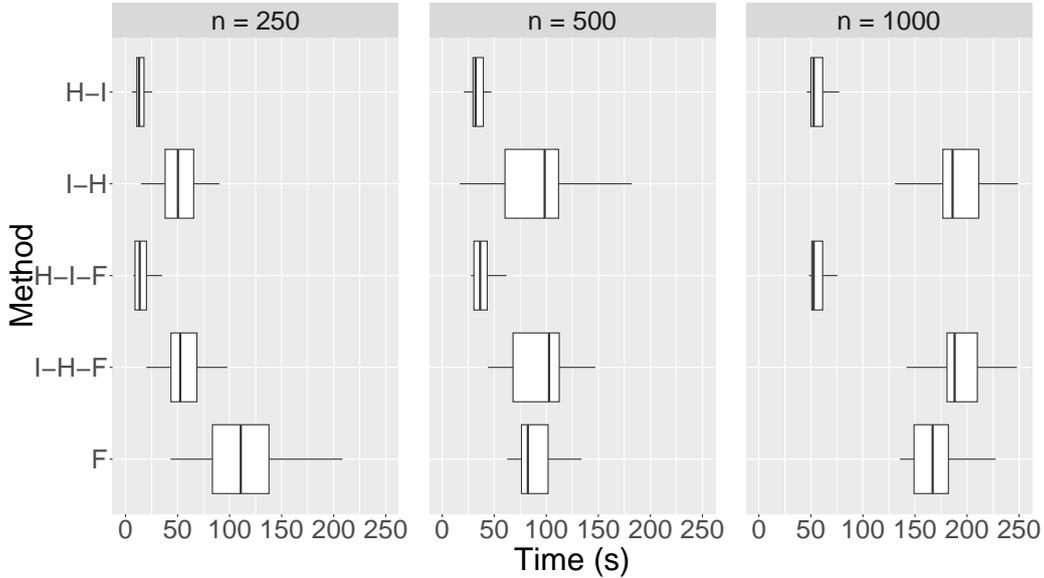


Figure 16: Simulation 1: boxplots of computational time (s) for each model selection approach and different values of n .

F Simulation 1: Combined Approach

Since H-I yields a higher probability of recovering the *input* layer than I-H, and I-H yields a higher probability of recovering the *hidden* layer than H-I, we have also considered the following approach: run both H-I and I-H independently, then take the input layer from H-I and the hidden layer from I-H to form the model; we refer to this as [H-I*]-[I-H*], where an asterisk denotes the layer being selected. We also investigate the aforementioned approach but followed by a fine-tuning phase, which we refer to as [H-I*]-[I-H*]-F. The simulation results for these procedures are given in Table 12.

We find that the [H-I*]-[I-H*] approach outperforms both the H-I and the I-H approaches, but is more computationally expensive. The addition of a fine-tuning phase improves the model selection performance further, but the proposed H-I-F approach has very similar performance while being much less computationally demanding.

Table 12: Model selection metrics for combined approaches.

n	Method	Time (s)	Input layer			Hidden layer		
			TNR	FDR	PI	\bar{q} (3)	PH	PT
250	[H-I*]-[I-H*]	44.96	0.83	0.20	0.61	2.89	0.42	0.25
500	[H-I*]-[I-H*]	131.50	0.90	0.11	0.79	3.19	0.81	0.64
1000	[H-I*]-[I-H*]	238.15	1.00	0.00	0.98	3.00	1.00	0.98
250	[H-I*]-[I-H*]-F	45.72	0.85	0.18	0.65	2.81	0.59	0.46
500	[H-I*]-[I-H*]-F	134.73	0.93	0.08	0.85	3.04	0.96	0.82
1000	[H-I*]-[I-H*]-F	240.06	1.00	0.01	0.99	3.00	1.00	0.98

Time (s), median time to completion in seconds (carried out on an Intel[®] Core[™] i5-10210U Processor).

G Simulation 2: Boxplots of TNR and q for Different Model Selection Objective Functions

This section contains the boxplots associated with TNR and q corresponding to Table 2 in Section 4.2 of the main paper.

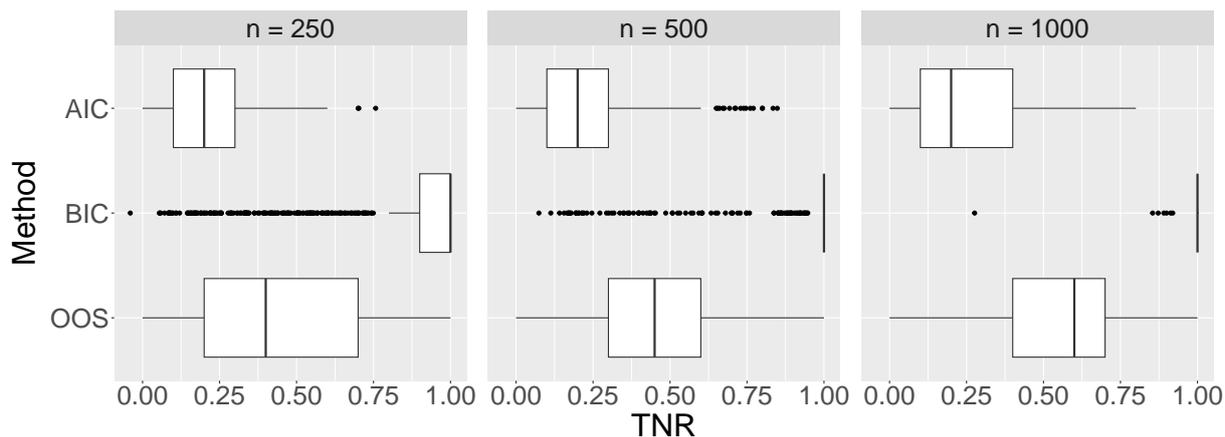


Figure 17: Simulation 2: boxplots for TNR (the true negative rate for the input variables) for the models selected by each objective function.

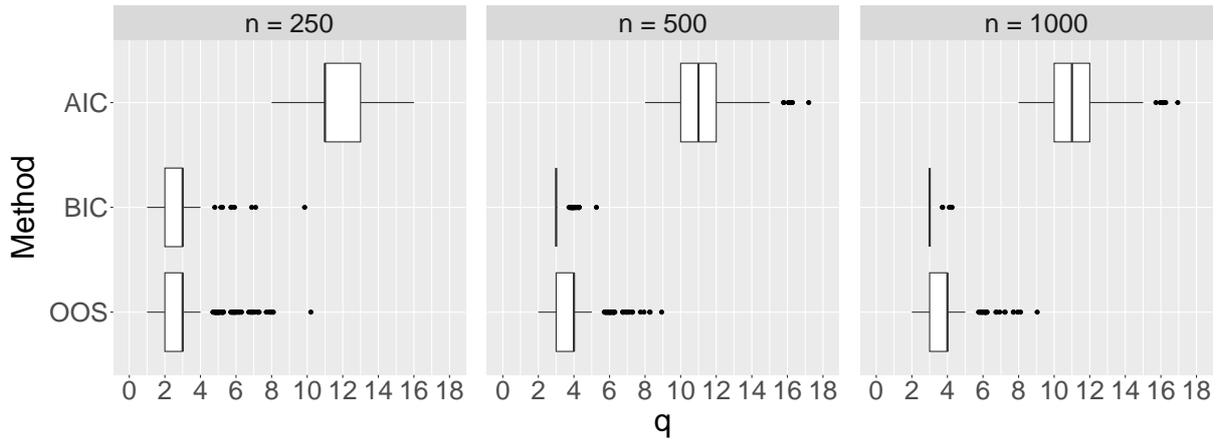


Figure 18: Simulation 2: boxplots for q (the number of hidden nodes selected) for the models selected by each objective function.

H Comparison of models selected with the full model trained using weight decay and early stopping

Over-parameterised neural networks can often suffer from issues of overfitting. Therefore, it is of interest to compare the out-of-sample performance of the models selected using our stepwise BIC procedure with the out-of-sample performance of the full model trained with common approaches that deal with overfitting. We investigate two popular approaches to overfitting: the use of a weight decay penalty, and early stopping. The weight decay penalty and the stopping point are both chosen based on the performance of the model on an additional validation data set (which is 20% the size of the training data set). Boxplots of the out-of-sample performance on the test data set for the BIC-selected model, the full model, and the models trained with weight decay and early stopping are displayed in Figure 19.

It is clear that both weight decay and early stopping improve the out-of-sample performance of the full model. However, the more parsimonious models that are selected via the proposed approach outperform both the commonly used weight decay and early stopping strategies.

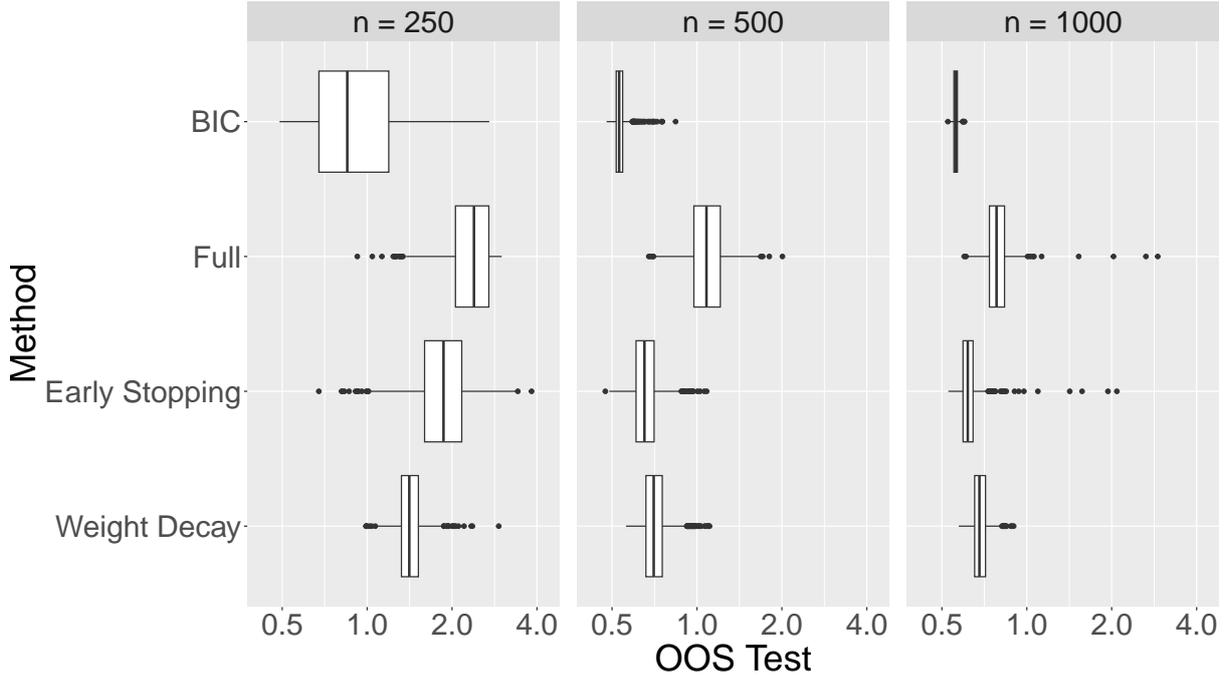


Figure 19: Boxplots of out-of-sample mean squared error evaluated on a test data set (on a logarithmic scale) for the models selected using the proposed approach, the full model, the full model with early stopping, and the full model with weight decay.

I ΔBIC and simple covariate effects

Simple measures of covariate importance and effects are used in Section 5 to accompany the model selection procedure for the data application. For the covariates that are selected, their relative importance is estimated using BIC differences (ΔBIC), which is given by $\Delta\text{BIC}_j = \text{BIC}_j - \text{BIC}_{\min}$, where BIC_{\min} is the BIC for the selected model (i.e., it is the model with the minimum BIC found by our algorithm) and BIC_j is the BIC for the model with the same hidden-layer structure as this selected model but with covariate j removed; hence, the more important covariate j is, the larger the corresponding ΔBIC_j value as its removal would lead to an increased BIC_j compared to BIC_{\min} . In addition to covariate importance, simple covariate effects ($\hat{\tau}_j$) are constructed by splitting the data into two groups based on whether or not the value of j th covariate for the i th individual, x_{ji} , is above or below the empirical median value, $m_j = \text{median}(x_{j1}, x_{j2}, \dots, x_{jn})$, and computing the difference in the average predicted response for these two groups. The corresponding equation is

$$\hat{\tau}_j = E[\text{NN}(X) \mid X_{(j)} > m_j] - E[\text{NN}(X) \mid X_{(j)} < m_j],$$

where $\text{NN}(X)$ is the output of the neural network for input covariate vector X (see Equation 2.1), $X_{(j)}$ denotes the j th covariate, and we take the conditional expected value

with respect to the empirical distribution of covariates in the dataset, i.e.,

$$E[\text{NN}(X) \mid X_{(j)} > m_j] = \frac{1}{\text{card}(i \mid x_{ji} > m_j)} \sum_{i \mid x_{ji} > m_j} \text{NN}(x_i)$$

where $\text{card}(\cdot)$ is the cardinality operator. Although this metric is a simplification of the (potentially non-linear) effects captured by the neural network, and our focus is on model selection, it is nevertheless a useful supplement to our approach that provides a high-level overview of the estimated covariate effects.