TU Delft

Delft University of Technology

‘Computer says no’ is not enough

Using prototypical examples to diagnose artificial neural networks for discrete choice analysis

Alwosheel, Ahmad; van Cranenburgh, Sander; Chorus, Caspar G.

DOI
10.1016/j.jocm.2019.100186

Publication date
2019

Document Version
Final published version

Published in
Journal of Choice Modelling

Citation (APA)
Alwosheel, A., van Cranenburgh, S., & Chorus, C. G. (2019). ‘Computer says no’ is not enough: Using prototypical examples to diagnose artificial neural networks for discrete choice analysis. *Journal of Choice Modelling*, *33*, Article 100186. https://doi.org/10.1016/j.jocm.2019.100186

Important note
To cite this publication, please use the final published version (if applicable).
Please check the document version above.

Copyright
Other than for strictly personal use, it is not permitted to download, forward or distribute the text or part of it, without the consent of the author(s) and/or copyright holder(s), unless the work is under an open content license such as Creative Commons.

Takedown policy
Please contact us and provide details if you believe this document breaches copyrights.
We will remove access to the work immediately and investigate your claim.

This work is downloaded from Delft University of Technology.
For technical reasons the number of authors shown on this cover page is limited to a maximum of 10.

Journal of Choice Modelling 33 (2019) 100186

Contents lists available at ScienceDirect

Journal of Choice Modelling

journal homepage: http://www.elsevier.com/locate/jocm

'Computer says no' is not enough: Using prototypical examples to diagnose artificial neural networks for discrete choice analysis

Ahmad Alwosheel *, Sander van Cranenburgh, Caspar G. Chorus

Transport and Logistics Group, Department of Engineering Systems and Services, Delft University of Technology, the Netherlands

A B S T R A C T

Artificial Neural Networks (ANNs) are increasingly used for discrete choice analysis, being appreciated in particular for their strong predictive power. However, many choice modellers are critical – and rightfully so – about using ANNs, for the reason that they are hard to diagnose. That is, for analysts it is hard to see whether a trained (estimated) ANN has learned intuitively reasonable relationships, as opposed to spurious, inexplicable or otherwise undesirable ones. As a result, choice modellers often find it difficult to trust an ANN, even if its predictive performance is strong. Inspired by research from the field of computer vision, this paper pioneers a low-cost and easy-to-implement methodology to diagnose ANNs in the context of choice behaviour analysis. The method involves synthesising prototypical examples after having trained the ANN. These prototypical examples expose the fundamental relationships that the ANN has learned. These, in turn, can be evaluated by the analyst to see whether they make sense and are desirable, or not. In this paper we show how to use such prototypical examples in the context of choice data and we discuss practical considerations for successfully diagnosing ANNs. Furthermore, we cross-validate our findings using techniques from traditional discrete choice analysis. Our results suggest that the proposed method helps build trust in well-functioning ANNs, and is able to flag poorly trained ANNs. As such, it helps choice modellers use ANNs for choice behaviour analysis in a more reliable and effective way.

1. Introduction

Throughout the choice modelling community there is a considerable and increasing interest in using Artificial Neural Networks (ANNs) to analyse and predict choice behaviour. Recently, several papers have emerged which show the added value of ANNs in a variety of settings (Alwosheel et al., 2018; Golshani et al., 2018; Lee et al., 2018; Sifringer et al., 2018; Van Cranenburgh and Alwosheel, 2019). This recent and rapid increase in interest can be explained by impressive achievements of ANNs in other fields, such as speech recognition and image classification. In particular, ANNs' flexible modelling structure and their ability to work with abundant, complex and highly non-linear data allow them to outperform statistical models (e.g., discrete choice models), most notably in their prediction accuracy (Hagenauer and Helbich, 2017; Karlaftis and Vlahogianni, 2011).

But, despite their often superior prediction performance, many choice modellers remain understandably critical towards ANNs. An important reason for this relates to their proverbial 'black box'-nature, meaning that the trained network itself is hard to interpret; this severely limits the behavioural insights that may be drawn from them, and makes them notoriously hard to diagnose (Karlaftis and Vlahogianni, 2011).[1] Regarding this latter aspect (model diagnosis), ANNs are typically validated by relying on empirical prediction performance. For instance, a widely-used validation approach is to evaluate ANN prediction performance on out-of-sample set of data points (Haykin, 2009). The error returned by the ANN on the out-of-sample set is an approximation of the so-called generalisation

* Corresponding author.
E-mail address: a.s.alwosheel@tudelft.nl (A. Alwosheel).

[1] In the machine learning community, training is equivalent to estimating in choice modellers' parlance, see Appendix 1 for more information on training ANNs.

https://doi.org/10.1016/j.jocm.2019.100186
Received 26 January 2019; Received in revised form 14 June 2019; Accepted 30 September 2019
Available online 8 October 2019
1755-5345/© 2019 The Author(s). Published by Elsevier Ltd. This is an open access article under the CC BY-NC-ND license (http://creativecommons.org/licenses/by-nc-nd/4.0/).

error, which is the key metric for assessing an ANN's learning capability. Having a low (high) generalisation error implies that the underlying data generating process has been well (poor) approximated (Abu-Mostafa et al., 2012; Breiman, 2001).

However, having a low generalisation error does not necessarily mean that the ANN has learned intuitively correct and desirable relationships. Due to the complexity of the learning task, in practice it occasionally happens that ANNs learn counterintuitive, inexplicable, or otherwise undesirable relations between variables. One evocative example in which an ANN learned undesirable relations occurred when it was used to select candidates in a recruitment context, and produced gender bias (i.e. favouring male candidates over female candidates) (Fernández and Fernández, 2019). While the ANN merely reproduced a bias that was implicitly present in the data on which it was trained, the opaqueness of the ANN made that such undesirable relations could stay unnoticed for a long time. Aside from this example, where an undesirable relation was learned by the ANN, in many cases a poorly trained network may have learned spurious or inexplicable relations between input and output data. This too, would create a problem in terms of the model's usefulness for making predictions, and it would not always surface in a process of merely evaluation prediction performance. In some fields trusting that the relationships between the explanatory variables and the predictions are reasonable, is more important than in others (Shmueli, 2010). In fields like natural language processing, the difficulty of diagnosing ANNs does not seem to substantially hinder applications; but, in the field of discrete choice behaviour analysis, having trust in the relations learned by a choice model, even when it is only used for predictions and not for behavioural or economic analysis, is rightfully considered very important.

Lately, the development of alternative techniques for validating and diagnosing ANNs in light of their black box-nature has been the subject of many debate, critiques, and research efforts in a variety of contexts (Lipton, 2016). Notably, in the computer vision field much research effort has recently been made (Montavon et al., 2018; Samek et al., 2017; Simonyan et al., 2013). A particularly interesting and easy-to-implement method that has been proposed in that field is based on the synthesis of so-called prototypical examples (Erhan et al., 2009; Montavon et al., 2018). By synthesising prototypical examples using the ANN (after having trained it), the ANN exposes the fundamental relationships that it has learned. These, in turn, can be evaluated by the analyst to see if they are intuitively correct and desirable, or not. For instance, when creating a prototypical example of a cat using an ANN that is trained to discriminate between cats and dogs (note that this involves 'drawing' a cat, as opposed to selecting a cat from the training data set), we expect to see distinguishable characteristics of cats in the prototypical example, such as whiskers. Hence, (human) interpretation of the prototypical example allows the analyst to diagnose the rationale behind the network predictions by comparing it with his (or her) mental image of a cat. To the best of authors' knowledge, no study has yet pioneered the use of prototypical examples to diagnose ANNs used for choice behaviour analysis.

This paper pioneers the use of prototypical examples to diagnose ANNs for choice behaviour analysis. In particular, we show that the generation and interpretation of prototypical examples is a low-cost and easy-to-implement method to validate the rationale of ANNs for choice behaviour analysis, and as such building trust (or not) in the ANN. To this aim, we first show that the use of prototypical examples for diagnosis is not confined to the domain of visual data, by re-conceptualising this notion towards one that is applicable for choice analysis. Subsequently, we apply this method to a recently collected Revealed Preference (RP) mode choice data set. That is, we train an ANN on these RP data and diagnose it by synthesising and interpreting prototypical examples. Finally, we cross-validate the proposed method, by comparing the relationships that are exposed by the prototypical examples with those obtained from traditional discrete choice models that were estimated on these same data.

Before we proceed, we would like to emphasize that this research effort does not aim to promote using ANNs for discrete choice behaviour analysis. In our view, since the interpretability of ANNs remains limited, even after having diagnosed it with the method proposed in this paper, the natural domain of application of ANNs is forecasting, rather than behavioural and economic analysis. This research effort is motivated by the increasing use of ANNs for discrete choice analysis (and prediction in particular), which in our view presents a strong motivation to offer a low-cost and easy-to-implement method to test whether the relationships learned by an ANN in the training process, are intuitive and desirable. Without such a diagnosis, a choice modeller remains unsure to what extent to trust the ANN, hampering its usefulness as a tool to make predictions.

The remainder of this paper is organised as follows: Section 2 introduces the prototypical example methodology to the choice modelling community. Section 3 provides the empirical case study. It presents the main dataset that we use for our analysis and it discusses the training of the ANN. Section 4 presents the prototypical examples created using our trained ANN. It shows how the methodology can be used to as a tool for diagnosing an ANN trained in the context of choice data. Section 5 provides a cross-validation of the proposed method, using conventional discrete choice analysis techniques. Finally, Section 6 draws conclusions and presents potential directions for future research. A first appendix presents a textbook-level introduction of how to train ANNs. The second appendix provides an additional case study, in which we show how the proposed prototyping method can be used to flag a poorly trained ANN.

2. Methodology

2.1. Model interpretability and diagnosis

Opening the black-box of ANNs has received much attention in a variety of fields. In the literature, several meanings have been attached to the effort of opening an ANN's black-box such as enhancing interpretability, explainability or understandability (Lipton, 2016). In this study, we use the term diagnosis which is closely related to the notion of interpretability; this latter concept is defined as the mapping of an abstract concept (e.g., a predicted mode choice) into a domain that the human can make sense of (Montavon et al., 2018). We consider the process of diagnosing an ANN to consist of the effort to test to what extent the ANN has learned intuitive and

desirable relations and hence can be trusted[2]; this is done by interpreting a set of synthesised examples (see further below), although it should be noticed here, that a full interpretation of the ANN is not the aim here.

Nonetheless, movement towards interpretable ANNs has created important tools that may (also) be used to diagnose ANNs (Ribeiro et al., 2016; Samek et al., 2017). This interpretability movement can be classified into ante-hoc and post-hoc approaches. In ante-hoc approaches interpretability is incorporated ('hard-wired') in the model structure. This approach entails designing a model such that its parameters and predictions can be interpreted by the analyst in a meaningful way. For choice modellers this approach is familiar, as the high level of interpretability of discrete choice models can be considered a result of imposing a strong structure as an ante-hoc approach. In other words, choice modellers routinely embed domain knowledge (e.g., theories on choice behaviour) into the model's structure. As a result, the parameters of, for instance, the widely used linear-additive Random Utility Maximisation (RUM) model (McFadden, 1973) can readily be interpreted as marginal utilities.

In post-hoc approaches interpretability comes after having trained a model (Montavon et al., 2018). Post-hoc methods provide bits of interpretation without elucidating or enforcing how the complete model works in full detail. Rather, they take a trained model and generate either a local (i.e., interpretation of a particular prediction) or a global interpretation of the model. Some post-hoc approaches have recently been applied to ANNs used in the choice modelling literature. Chiang et al. (2006) and Hagenauer and Helbich (2017) conduct sensitivity analysis to measure the importance of input variables for different types of trained ANNs. Golshani et al. (2018) implemented Garson's algorithm (Garson, 1991), which aims to determine the relative importance of input attributes, for explaining the ANN's predictions. Note that a recent study highlights the drawbacks of using sensitivity analysis for interpretability (Samek et al., 2017).

2.2. Synthesising prototypical examples for diagnosing an ANN

Synthesising prototypical examples is another post-hoc approach to provide interpretation to ANNs. It has been proposed in the computer vision field, where it is difficult to understand exactly how a trained ANN functions due to the large number of interacting and non-linear parts (e.g., an ANN called AlexNet consists of over 60 million parameters) (Krizhevsky et al., 2012). Synthesising prototypical examples is considered a low-cost method to uncover a sample of the learned patterns in images (Erhan et al., 2009; Simonyan et al., 2013). The idea of this approach is that by synthesising prototypical examples using a trained ANN (note this means drawing an image, not selecting one from a data set), these prototypes expose fundamental relationships that the trained ANN has learned. As such, the analyst can assess the synthetically generated prototypes to see if they make sense, e.g. contain the expected characteristics. Hence, prototypical examples allow the analyst to diagnose part of the rationale behind the network predictions and build trust on the predictions by comparing it with his or her own mental map. Note that because the set of generated prototypical examples is generally small compared to the number of relations learned by the network, it is not able to generate a complete interpretation of the ANN; nonetheless, its use for diagnosing and building trust in ANNs is well established in the field of computer vision.

2.2.1. Activation maximisation

To synthesise prototypes, a technique called activation maximisation is used. This technique searches for the input that maximises the probability of a particular output label (e.g. a cat or a dog). The inputs that maximise the probability of a certain output label are called 'prototypical examples'; these examples, in the context of image classification, are synthetic drawings of a dog or a cat, rather than a particular image of a dog or a cat selected from the input data. The process of maximising the activation is much akin to the process of training an ANN (a detailed description of this process is presented in the Appendix for interested readers). In the context of choice data, the set of observations is given by $S = ((\mathbf{x}_1,\mathbf{y}_1),(\mathbf{x}_2,\mathbf{y}_2),\ldots,(\mathbf{x}_n,\mathbf{y}_n),\ldots,(\mathbf{x}_N,\mathbf{y}_N))$. Each nth observation s_n contains a vector of independent variables $\mathbf{x}_n$ that represent the attributes of alternatives, socio-demographics, and possibly other covariates, and a K-dimensional vector of dependent variables $\mathbf{y}_n$ that represent the observed choice (i.e., zeros for non-chosen alternatives, and a one for the chosen alternative); K being the size of the choice set. The ANN training process aims to find the optimum weight parameters $\mathbf{w}$ such that the error function (which depends on the observations' variables $\mathbf{x}$ and $\mathbf{y}$) is minimised.

To maximise the activation within a trained ANN, we aim to find input values (i.e., the prototype) $\mathbf{x}^*$ such that the activation of a particular output neuron (i.e., the choice probability P_i for alternative i) is maximised (equation (1))

$$\mathbf{x}^* = \underset{\mathbf{x}}{\operatorname{argmax}}(P_i(\mathbf{w},\mathbf{x}) - \lambda\,\mathbf{x}^*) \tag{1}$$

where λ represents the regularisation term.[3]

Like the process of training an ANN, the process of finding $\mathbf{x}^*$ is iterative. At the beginning, the $\mathbf{x}$ values are initialised (either randomly or from some predefined point). Then, at each iteration step, P is increased by changing the $\mathbf{x}$ (note that the weights $\mathbf{w}$ are kept fixed during this process). A gradient ascent approach can be used for this purpose (Erhan et al., 2009). This process is repeated until a pre-specified stopping criterion is achieved. Note that equation (1) is non-convex, meaning that there are many local maxima in the search space. As such, it is possible to obtain several solutions, meaning that a single ANN can generate many different prototypical examples.

[2] Note that a growing body of literature studies what constitutes a trustworthy model (Kulesza et al., 2015; Lipton, 2016; Miller, 2017).

[3] Regularisation is a frequently used method in machine learning to avoid overfitting by penalising the size the models flexibility (Bishop, 2006).

Fig. 1. Left: Actual dumbbell. Right: Four prototypical examples dumbbells (Mordvintsev et al., 2015).

In case **x** is randomly initialised, the gradient ascent process usually is able to produce many prototypical examples. However, previous studies have shown that many of them can be useless, in the sense that they do not resemble meaningful inputs **x** (e.g. because they take extreme and unrealistic values (Nguyen et al., 2015)). To overcome this problem, the optimisation process can be constrained to only generate examples that resemble realistic inputs (Nguyen et al., 2016a). In particular, two controlling factors can be applied: First, instead of randomly initialising **x** values (as was initially proposed by Erhan et al. (2009)), **x** can be initialised based on the mean and variance of the attributes levels as they appear in the original data (Nguyen et al., 2016b). This is found to substantially improve the quality of the synthesised prototypes. Second, **x** values can be constrained to be within the range of the original data. In the field of image processing this practice is widely used (called clipping, see Yosinski et al. (2015) for example). A pseudocode of the activation maximisation method can be found below.

Pseudocode: Activation Maximisation Method

Step 1 Initialisation
  Set **x** values (according to the selected initialisation method)
Step 2 Forward propagation
  Propagate the input neuron values **x** to output neuron through hidden neurons
  Calculate the ANN output neuron values (ANN probabilities)
Step 3 Backward propagation
  Calculate the gradient for the network input neurons
  Update **x** values (within the range of the original data)
  Go back to step 2 and repeat the process until the selected criterion is satisfied

2.3. Prototypical examples - a computer vision illustration

To further clarify the method, in this subsection we provide an illustration of how prototypical examples are used in the computer vision field. An illustration from the computer vision field is taken for two reasons: 1) many of the recent advancements in ANN research have taken place in this field; 2) visual examples are particularly effective in illustrating how the method works. The particular example is taken from Mordvintsev et al. (2015). Mordvintsev et al. (2015) use an ANN to discriminate between several output classes, including a dumbbell (weight) class. To train this ANN, they presented the ANN with many images, including many images of dumbbells. To check whether the trained network has learned the relevant features of dumbbells (e.g., a dumbbell has a bar and weight plates) and ignores unrelated ones (e.g., weight plates can be of various shapes), prototypes of dumbbells were synthesised.

Fig. 1 shows four different prototypical examples of dumbbells generated by the trained ANN. From Fig. 1 we can make a number of observations. First we notice that the four prototypes are not the same. But it is clear that they all present patterns that, once interpreted, can be recognised by the human analyst as dumbbells. Second, the prototypes always show a part of a muscular weightlifter's arm. Depending on the analyst expectations,[4] this may either indicate that the trained network has failed to completely learn the features of a dumbbell (as it is mixed with the arm of a muscular weightlifter). Hence, the predictions of the trained network may not be fully trusted, as the trained ANN may confuse muscular weightlifter images with dumbbell images. A possible source of this 'failure' is that the examples used for training contain dumbbells and arms holding them. Most importantly for the purpose of our paper, this example illustrates that prototypical examples can be used to diagnose the rationale of a trained ANN.

3. Data and ANN training

3.1. Data preparation

For this study, we use revealed preference (RP) data from a study conducted for travel mode choice analysis in London city (Hillel

[4] Analysts may have different beliefs and expectations. For example, some would expect (and accept) that the muscular weightlifter appears in dumbbells prototypes, whereas others may not. Most importantly, the synthesised prototypes can be used to reveal these relations learned by the ANN (either expected or not), and as such help the analyst to determine whether or not an analyst will trust the ANN.

Table 1
Data statistics.

No.	Attribute	Description	Type	Range [min, max]	Mean and standard deviation
1	TC_{Drive}	Estimated cost of driving route, including fuel cost and congestion charge	Float (£)	[0.05, 17.16]	(1.91, 3.48)
2	TC_{PubTr}	Estimated cost of public transport route, accounting for rail and bus fare types	Float (£)	[0, 13.49]	(1.56, 1.55)
3	TT_{Drive}	Predicted duration of driving route	Float (hours)	[0.03, 2.06]	(0.29, 0.25)
4	TT_{PubTr}	Predicted duration of public transportation	Float (hours)	[0.03, 2.73]	(0.47, 0.31)
5	TT_{Walk}	Predicted total duration of walking times for interchanges on public transport route	Float (hours)	[0.04, 9.27]	(1.15, 1.13)
6	*DIS*	Straight line distance between origin and destination	Integer (meters)	[96, 40941]	(4690, 4827)
7	*TRAF*	Predicted traffic variability on driving route	Float	[0, 1]	(0.34, 0.20)
8	*INTER*	Number of interchanges on public transport route from directions API	Integer	[0, 4]	(0.38, 0.62)
9	*DL*	Boolean identifier of a person making trip: 1 if person has driving license, 0 otherwise	Bool	[0, 1]	(0.62, 0.49)
10	*CO*	Car ownership of household person belongs to: no cars in household (0), less than one car per adult (1), one or more cars per adult (2)	Integer	[0, 2]	(0.99, 0.75)
11	*BS*	Bus fare scale of person making trip imputed from socio-economic data: 0 (free bus journeys), 0.5 (half price), 1 (full price)	Float	[0, 1]	(0.64, 0.47)
12	*FEM*	Boolean identifier of a person making trip: 1 if female, 0 otherwise	Bool	[0, 1]	(0.53, 0.49)
13	*AG*	Age of person making trip	Integer (years)	[5, 99]	(39.5, 19.3)

et al., 2018).,[56] The chosen dataset contains four alternatives and a total of 27 features (i.e., attributes of alternatives and characteristics of decision makers). To prepare these data for our study, we took a number of steps. First, we removed features that were deemed redundant and merged them with others. For instance, rather than using three features to represent car cost (fuel, congestion, and total cost), we merged them into one total cost feature. Table 1 presents statistics on the attribute levels in the data set used for analysis. Second, we noticed that the data set was highly imbalanced in terms of chosen mode distribution: walking (17.6%), cycling (3.0%), public transport (35.3%) and driving (44.2%). Such imbalances could potentially be problematic for training ANNs (Haykin, 2009). In light of the aim of this paper (which does not focus on finding the best predictions of travel behaviour, but rather on testing a method to diagnose the ANN), we deemed dealing with this sort of data imbalances out of scope. Therefore, we 'repaired' this data imbalance by removing the cycling alternative from the data set. However, we consider exploring the use of prototypical examples in the context of imbalanced data sets an interesting avenue for further research. Third, we excluded very short trips (i.e., less than two minutes), as these were deemed not to contain a mode trade-off. The resulting dataset that is used for this study consist of 77,638 mode choice observations.

3.2. ANN development and training

The ANN is implemented in the Python environment using the open source deep learning library Keras (Chollet, 2015).[7] To train the ANN, the built-in training algorithm (which is used to update weights' values $\mathbf{w}$) known as Adam is used (Kingma and Ba, 2014).[8] Prior to training the ANN, the data are normalised to reduce training time and minimise the probability of ending-up with suboptimal solutions.[9] A conventional three layers (input, output and one hidden layer) fully connected ANN structure is used (the used ANN has $M = 173$; M being the number of adjustable parameters).[10] To test the performance of the ANN to predict the travel mode choice, we conducted a so-called k-fold cross-validation method, with $k = 5$. The dataset is partitioned into five equally sized folds of (roughly) 15, 528 observations. Then, a single fold is used for testing, while the remaining four folds are used for training. This process is repeated 5

[5] The dataset and its description are available online, and can be downloaded from the first author profile at researchgate.net.

[6] In addition, we use the well-known stated preference Swiss Metro data, as a second case study, to take a first step towards testing the general applicability of the proposed method. To avoid repetition in the main text, we present this second case study in Appendix 2.

[7] Code is available upon request from the first author.

[8] Note that the used training algorithm is more sophisticated than the described training process at the Appendix (i.e., the Adam algorithm implements a version of stochastic gradient descent), but both are based on a backpropagation training mechanism.

[9] Data normalisation is common practice for ANN training. In this study, the minimum and maximum values of data are normalised to −1 to +1.

[10] ANN complexity is adjusted using a cross validation approach (see (Alwosheel et al., 2018) for more details). To avoid overfitting, a commonly used rule-of-thumb in ANNs is that the sample size needs to be (at least) 10 times larger than the number of adjustable parameters in the network (Haykin, 2009). A recent study specifically dealing with sample size requirements for using ANNs in the context of choice models is more conservative, and recommends to use a sample size of (at least) 50 times the number of estimable weights (Alwosheel et al., 2018). Our sample size satisfies this condition and, therefore, we safely avoid overfitting issues.

Table 2
Performance of the trained ANN.

Performance metric	Function	Null model	ANN	Linear-additive RUM
Final Log-likelihood	$\sum_{n=1}^{N}\sum_{k=1}^{K} y_{nk}\ln(P_{nk})$	−86,625	−43,477	−50,704
Cross-entropy	$\frac{1}{N}\sum_{n=1}^{N}\sum_{k=1}^{K} y_{nk}\ln(P_{nk})$	−1.09	−0.56	−0.65
ρ^2	$\rho = 1 - \frac{LL(\widehat{\beta})}{LL(0)}$	0	0.50	0.42

times, where each of the five folds is used only once for testing.

Table 2 shows several performance metrics for the trained ANN. The reported performance metrics are averaged across the five folds. It shows that ANN achieves a fair prediction performance. For comparison, we also report the performance of a standard linear-additive RUM-MNL model.[11] As expected based on previous literature, the ANN outperforms the discrete choice model by a large margin. Table 3 shows the *k*-folds confusion matrix for the trained ANN. To construct the confusion matrix each observation is assigned to an alternative based on the highest probability as predicted by the ANN. Then, each prediction is compared to the true chosen alternative. The cells on the diagonal show the mean percentage of the observations that are correctly assigned, across the 5 folds. Additionally, the values between parentheses show the average ANN probabilities of the observations that are correctly classified. The off-diagonal cells show the mean percentage of observations that are misclassified, across the 5 folds. Values between parentheses show the average ANN probabilities of these observations. Table 3 shows that the ANN predicts driving choices quite well, while it has some more difficulty with accurately predicting choices for walking and public transport. However, it should be noticed that there can be different degrees of randomness associated with choices for different modes. Hence, it could be the case that it is inherently more difficult to accurately predict choices for public transport than for driving. Another possible explanation could be related to imbalances in the training data (i.e., the driving alternative has 45% share of the data, as mentioned above). It is well known, that under-representation of particular choice alternatives ('output labels' in machine learning parlance) can undermine the reliability of a trained ANN model. Dealing with such problems is beyond the interest of this work, but a plethora of methods and approaches are developed to combat these issues. For example, a commonly used approach is to synthesise more observations of the under-represented alternative (see e.g. Chawla et al. (2002)). Another approach is based on penalising the ANN when it makes classification mistakes concerning under-represented alternatives. For further discussions on these methods, interested readers are referred to e.g., He and Garcia (2008) and Batista et al. (2004).

4. Results: prototypical examples

Now that we know that, as expected, the ANN greatly outperforms a conventional discrete choice model in terms of predictive power, the big question becomes: can we trust that the predictive power of ANN is based on intuitive, explicable and desirable relations between in- and output variables which the network has learned (or not)? Such a conclusion cannot be drawn by simply inspecting prediction outcomes such as the ones presented above; and this is where the proposed method of creating prototypical examples comes in. Using the techniques described in Section 2, we let the trained ANN create typical examples of a choice for a car (driver), for public transport, and for walking. Inspecting of these synthetic examples allows us to determine to what extent we can trust the ANN. Table 4 shows, for each alternative (driving, public transport, walking), five prototypical examples created by the trained ANN. Note that each example is independently synthesised. That is, the initial inputs are independently initialised. To facilitate inspection, we employ a so-called vertical heat-map, where high values are depicted red and low values are depicted blue. For example, the red colour at the last attribute 'age' (AG) indicates high values (between 50 and 53), while blue indicates low values.

A number of inferences can be made based on Table 4. First, we use the examples to diagnose whether our ANN has learned intuitively correct relationships, as opposed to spurious or otherwise undesirable ones. For instance, Table 4 shows that prototypical examples in which a travel mode is chosen are associated with relatively low travel times for that mode (columns 3 to 5). For instance, in the prototypical examples of a choice to drive, driving has the lowest travel time of available modes. Likewise, we see that a prototypical choice to drive is associated with a high number of interchanges for the public transport alternative (column 8), a high number of owned cars (column 10) and a low variability in traffic conditions (column 7). Furthermore, the prototypical case in which walking is chosen is associated with a relatively low distance to the destination. All these results are all in line with expectations. Second, the synthesised prototypes resemble realistic relations. For instance, distances for driving prototypes are between 13 and 16 km, while they are around 200 m for walking prototypes. Furthermore, a prototypical driving trip takes about 15 min (0.25 h), and a prototypical walking trip takes less than five minutes. Third, it should be noted that the prototypes may also show some unexpected patterns. For example, prototypes for driving are expected to show that a person has a driving license, but they do not. Instead, prototypes for public transport are associated with high driving license ownership. Although this is not as expected, finding such

[11] Note that the *k*-fold method is not used for the RUM model. Rather, the RUM model is estimated one time using the whole dataset.
[12] For optimal reading, this table is best shown in screen and coloured printing.

Table 3
Confusion matrix.

		ANN Classification			
		Driving	Public Transport	Walking	∑
True chosen alternative	Driving	83.45 (0.68)	10.85 (0.20)	5.68 (0.12)	100% (1)
	Public Transport	21.6 (0.23)	71.52 (0.67)	6.7 (0.10)	100% (1)
	Walking	21.2 (0.24)	9.5 (0.20)	69.3 (0.56)	100% (1)

Table 4
Synthesised prototypical examples. See Table 1 for description of the attributes.[12]

Attribute No.		1	2	3	4	5	6	7	8	9	10	11	12	13
		Travel Cost		Travel time			Trip characteristics			Traveller characteritics				
		TC_{Driv}	TC_{Pubtr}	TT_{Driv}	TT_{Pubtr}	TT_{Walk}	DIS	TRAF	INTER	DL	CO	BS	FEM	AG
Driving	Ex. 1	6.42	3.81	0.26	2.51	5.15	15953	0.43	2.24	0.41	1.16	0.50	0.52	52
	Ex. 2	6.40	3.70	0.25	2.47	5.15	15381	0.42	2.23	0.42	1.16	0.55	0.55	51
	Ex. 3	6.34	3.74	0.24	2.49	4.90	14249	0.41	2.17	0.46	1.17	0.53	0.51	50
	Ex. 4	6.25	3.79	0.22	2.51	4.65	13006	0.43	2.08	0.52	1.18	0.53	0.54	48
	Ex. 5	6.37	3.63	0.28	2.50	5.27	15919	0.43	2.29	0.45	1.16	0.54	0.56	53
Public Tran	Ex. 1	8.94	7.78	1.37	0.92	3.12	17524	0.54	1.77	0.57	0.68	0.51	0.49	48
	Ex. 2	9.20	7.76	1.35	0.90	3.20	16756	0.54	1.66	0.54	0.68	0.50	0.46	49
	Ex. 3	9.47	7.60	1.34	0.89	3.18	16365	0.53	1.62	0.52	0.69	0.53	0.50	48
	Ex. 4	9.13	7.74	1.31	0.89	3.09	17001	0.56	1.69	0.55	0.67	0.48	0.53	48
	Ex. 5	9.01	7.79	1.35	0.91	3.06	17081	0.53	1.61	0.52	0.69	0.53	0.53	48
Walking	Ex. 1	8.40	8.75	0.52	1.60	0.07	198	0.49	1.69	0.46	0.72	0.47	0.53	40
	Ex. 2	8.79	8.68	0.53	1.63	0.07	179	0.47	1.70	0.46	0.72	0.45	0.51	41
	Ex. 3	8.36	8.78	0.51	1.58	0.06	181	0.50	1.67	0.45	0.71	0.51	0.48	41
	Ex. 4	8.41	8.82	0.51	1.58	0.06	199	0.51	1.72	0.51	0.70	0.50	0.54	41
	Ex. 5	8.66	8.82	0.50	1.60	0.06	198	0.53	1.65	0.44	0.74	0.48	0.45	39

relations can be useful for understanding the data and the model. For instance, they could indicate that an attribute is not relevant for explaining predictions, or could point towards issues related to the data itself. For example, after seeing this result, we found that the driving alternative also includes car passenger, taxi, van and motor bike (see Hillel et al. (2018) for more information). This could explain why prototypical examples for driving alternative do not associate with driving license ownership.

Finally, as alluded to in section 2.2.1 for this method setting λ to an appropriate value is needed. In particular, we find that in case λ is set too small the regularisation term will have little effect. As a result, the generated prototypical examples may have extreme attribute levels, while in case λ is set too large the prototypical examples will contain only noise. In this study, we have tested numerous values for λs and found $\lambda = 0.005$ to work best (in the sense that it resulted in the generation of prototypes that are meaningful, i.e. from the perspective of the analyst). This result is in line with work conducted in computer vision, where also $\lambda = 0.005$ is found to work well, see e.g. Nguyen et al. (2016a).[13] As such, we recommend using this value in future work.

5. Cross-validation using discrete choice models

This section cross-validates the results and interpretations presented in section 4. To do so, we compare our findings with results obtained from a traditional linear-additive RUM-MNL discrete choice model. More specifically, we use this estimated conventional discrete choice model for two purposes: First, to inspect whether the synthesised prototypical examples would also be considered realistic prototypes from the discrete choice model's perspective. In other words, do the prototypical examples obtain high prediction probabilities when fed into the choice model? Second, we use this choice model to put the derived interpretations (in section 4) to the test. Since the estimates of discrete choice models allow for straightforward inference of the importance of attributes, we can use them as a test to check the interpretations on attribute importance derived from the synthesised prototypes. It is very important to note here, that we do not aim to compare the trained ANN and the estimated discrete choice model (DCM) in terms of their interpretability; clearly, the DCM beats the ANN in this regard, given its strong and behaviourally intuitive model structure which facilitates rigorous behavioural and economic interpretation. Our aim is different: given the strong predictive power of the ANN (compared to DCM), we use prototypical examples to diagnose and build trust in the model (which is a more modest aim than achieving a full interpretation); in this section we validate our diagnosis method by checking whether the generated prototypical examples are congruent with the

[13] Also note that we use the same regularisation value for the additional case study we show in Appendix 2.

Table 5
Estimation results of RUM-MNL model.[15]

No. of observations	77,638
Final LL	−50,704.14
ρ^2	0.42

Attribute	*Est.*	*Rob. t-values*	*Part-worth utility*
ASC_Drive	0	fixed	
ASC_PubTr	1.85	59.57	
ASC_Walk	2.62	73.91	
TT	−6.11	−95.48	4.30
TC	−0.121	−7.94	0.27
DL	0.994	44.23	0.994
BS	−0.112	−4.73	0.112
CO	1.39	90.38	2.78
INTER	0.767	38.02	0.767
TRAF	−2.77	−43.00	1.04
AG_TC	−0.121	−3.92	0.09
DIS_TC	0.00840	9.75	0.12
FEM_TC	−0.0348	−4.08	0.05

estimated DCM, whose model structure we trust a priori.

Table 5 shows the estimation results of the DCM alongside the implied part-worth utilities.[14] As can be seen, and as is expected, all parameters have the intuitively correct sign and are highly significantly different from zero (see Appendix 3 for the model specifications). More interestingly, Table 6 shows that the prototypes created by the ANN can also be considered prototypes from the discrete choice model's perspective. That is, we see that the all prototypes yield very high choice probabilities from the estimated choice model's perspective. This result validates the synthesised examples. Furthermore, Table 5 confirms the interpretations derived in section 4 regarding attribute importance. Specifically, both the prototypes and the part-worth utilities indicate that travel time is a highly important factor the mode choice (i.e., prototypical examples are always associated with the lowest travel time). Additionally, the importance of car ownership and light traffic conditions for the driving alternative are cross-validated by respectively the 2nd and 3rd largest associated part-worth utilities.

6. Conclusions and recommendations

This study contributes to the growing literature which focuses on using machine learning techniques for choice behaviour analysis, by pioneering a post-hoc methodology for diagnosing trained ANNs (in the context of choice behaviour analysis). We show how the proposed methodology can be easily applied at low cost to build trust in an ANN which was trained to predict (mode) choice behaviour. Based on our encouraging results, we believe that the proposed methodology provides a valuable tool for discrete choice modellers. It is however crucial to mention once more that the proposed method does not entirely open-up the black box of an ANN. As such, in our view of the most natural domain of application of ANNs still lies in forecasting tasks, as opposed to behavioural or economic analysis; our prototypical examples method helps the analyst to determine whether or not to trust predictions made by the trained ANN.

We would like to point out several limitations to this study, providing avenues for future research. Firstly, to avoid synthesising unrealistic prototypical examples, in our study the prototypical examples are randomly initialised according to a pre-defined distribution (normal distribution). In future research, a more accurate initialisation process can be employed. In particular, incorporating generative models (e.g., generative adversarial network as proposed in Goodfellow et al. (2014)) in the initialisation process may produce more reliable prototypes. Secondly, the empirical analyses provided in this paper are based on two datasets (one of which is presented in Appendix 2). It is advisable to repeat these type of analyses using more datasets having different characteristics, e.g. many attributes, more alternatives, data imbalances, etc.. This will provide a richer view on the extent to which the proposed 'prototypical examples' methodology is a valuable tool to diagnose trained ANNs more generally. Lastly, the method is based on an inherently subjective process on the side of the analyst, when deriving interpretations from synthesised prototypes; that is, the analyst ultimately decided to what extent the generated examples match his or her expectations regarding the phenomenon being modelled (e.g. mode choices). Although we believe that this is certainly not a disadvantage per se, it is worthwhile to develop more objective methods or guidelines to extract and interpret prototypes. This will improve the rigour of this method and ultimately will help analysts to better understand the potential and limitations of using ANNs for discrete choice analysis.

[14] To calculate the part-worth utility the attribute level range is multiplied by parameter estimation. However, since we work with RP data here we use the 20–80 percentile range (as opposed to the full range). For example, consider an attribute A that consists of uniformly distributed values between 0 and 100. From that, we get the 20–80 percentile range (60 in this case) and multiply it by the parameter estimate.

[15] Note that the model is estimated using the whole dataset (*k*-folds method is not used in this case).

Table 6
Choice probabilities for prototypical examples, based on discrete choice model.

Alternatives	Examples No.	P_{Drive}	P_{PubTr}	P_{Walk}
Alt. 1: Driving	Example 1	0.9999	0.0000	0.0000
	Example 2	0.9999	0.0000	0.0000
	Example 3	0.9999	0.0000	0.0000
	Example 4	0.9999	0.0000	0.0000
	Example 5	0.9999	0.0000	0.0000
Alt. 2: Public Transport	Example 1	0.0025	0.9974	0.0000
	Example 2	0.0027	0.9972	0.0000
	Example 3	0.0027	0.9972	0.0000
	Example 4	0.0030	0.9969	0.0000
	Example 5	0.0032	0.9967	0.0000
Alt. 3: Walking	Example 1	0.0010	0.0000	0.9989
	Example 2	0.0009	0.0000	0.9990
	Example 3	0.0010	0.0000	0.9988
	Example 4	0.0010	0.0000	0.9988
	Example 5	0.0010	0.0000	0.9989

Statement of contribution

Artificial Neural Networks (ANNs) are increasingly being used to analyse choice behaviour. They usually outperform their theory-driven counterparts, i.e. traditional discrete choice models, in terms of prediction performance. Nonetheless, many choice modellers are reluctant to use ANNs for choice behaviour analysis. An important reason for this is that ANNs are hard to diagnose (black-box issue). That is, for analysts it is hard to see whether an ANN has learned intuitively correct relationships, as opposed to spurious or otherwise undesirable ones. This paper is the first to investigate the ANN black-box issue in the context of choice behaviour analysis.

It contributes to the literature by pioneering a methodology that can be used to diagnose the rationale behind the predictions of ANNs in the context of choice behaviour analysis. It does so by re-conceptualising a method that has been proposed in the computer vision context toward one that is applicable for choice analysis. The method is based on the creation of so-called prototypical examples. By letting a trained ANN synthesise prototypical examples, the ANN reveals the fundamental relationships that it has learned. These, in turn, can be assessed by the analyst to derive interpretations regarding the rationale of the trained ANN.

Based on the encouraging interpretations we have obtained, we believe that the proposed methodology provides a valuable tool for discrete choice modellers, as it allows for a test of trustworthiness of the otherwise black box ANN.

Acknowledgment

We are grateful to three anonymous reviewers for their helpful suggestions on an earlier version of this manuscript. The first author would like to thank King Abdulaziz City for Science and Technology (KACST) for supporting this work. The third author would like to acknowledge funding from the ERC, consolidator grant BEHAVE - 724431.

Appedix 1. Training of ANNs

ANNs are biologically inspired systems that have proven to be a powerful technique for machine learning. They are well-known for being highly effective in solving complex classification and regression problems (Haykin, 2009). ANNs consist of highly interconnected processing elements, called neurons, which communicate together to perform a learning task, such as classification, based on a set of observations. There are three types of neurons: input neurons, hidden neurons, and output neurons. Input neurons represent the independent variables. In the context of choice modelling, these are the alternatives' attributes, characteristics of decision-makers, and contextual factors. Output neurons contain the dependent variables. In a discrete choice context, these are the choice probabilities for each alternative. Neurons in-between are called hidden neurons because their inputs and outputs are connected to other neurons and are therefore 'invisible' to the analyst.

Each neuron in the network (except input neurons) receives inputs multiplied by estimable parameters known as weights (w_i). The weighted inputs are accumulated and added to a constant (called bias) to form a single input for a pre-defined processing function known as activation function. The bias has the effect of increasing or decreasing the net input of the activation function by a constant value, which increases the ANNs flexibility (Haykin, 2009). The activation function generates one output that is fanned out to other neurons. Commonly used activation functions are tan-sigmoid, softmax and step function.

In this work, we use the widely implemented ANN structure that consists of layers of neurons connected successively, known as multilayer perceptron (MLP) structure. Three elements need to be defined for MLP structure: activation functions, number of layers, and number of neurons at each layer. These three elements are set according to the desired objective of the modelling effort. For example, adding more layers or more hidden neurons increase the complexity of the network. In this paper, we use the so-called shallow version of ANN (i.e., an ANN with input layer, output layer, and one hidden layer). The complexity of which is adjusted by modifying the number of hidden neurons (we found that 20 hidden neurons provides satisfactory performance). The activation

functions employed in this paper are tang-sigmoid softmax function for hidden and output neurons, respectively.

Once these three elements are defined, the training process that aims to find the model's parameter $(\mathbf{w})$ is implemented. The choice data that we use for training consist of set of observation observations $S = ((\mathbf{x}_1,\mathbf{y}_1),(\mathbf{x}_2,\mathbf{y}_2),...,(\mathbf{x}_n,\mathbf{y}_n),...,(\mathbf{x}_N,\mathbf{y}_N))$. Each nth observation s_n contains a vector of independent variables $\mathbf{x}_n$ that represent the attributes and a K-dimensional vector of dependent variables $\mathbf{y}_n$ that represent the observed choice (i.e., zeros for the non-chosen alternatives, and a one for the chosen alternative); K being the size of the choice set. Since choices are mutually exclusive (i.e., only one alternative can be chosen from the choice set), from a machine learning perspective this is considered a classification problem.

The central goal of training is to model the underlying data generating process (DGP) that has led to the current set of observations, so that the best possible prediction for future observation is achieved (Bishop, 1995). While to estimate the parameter of a choice model the likelihood function is maximised, for ANN training an equivalent so-called error function $J(\mathbf{w})$ is minimised. We define $\mathbf{w}$ as a vector that contains the ANN estimable parameters w. Assuming the data consist of N choice observations across K alternatives, the error function is defined as follows:

$$J(\mathbf{w}) = -\sum_{n=1}^{N}\sum_{k=1}^{K} y_{nk}\ln(P_{nk}) \tag{A1}$$

Where y_{nk} is an indicator which denotes whether alternative k is chosen in observation n, and P_{nk} is the choice probability predicted by the ANN, which is a function of $\mathbf{w}$ and $\mathbf{x}$. By training the ANN, the analyst's objective is to find the weight vector $\mathbf{w}$ such that $J(\mathbf{w})$ is minimised. This process can be described as follows:

$$\mathbf{w}^* = \underset{\mathbf{w}}{\operatorname{argmin}}\ J(\mathbf{w}) \tag{A2}$$

The process of finding optimum weights $(\mathbf{w}^*)$ is conducted in successive steps. At each step, $J(\mathbf{w})$ is decreased by adjusting the parameters in $\mathbf{w}$. The well-known gradient descent approach is the widely applied algorithm for this purpose.[16] In short, this process of training an ANN can be described as follows: first, the weights' values w are randomly initialised. The input neurons' values (taken from the training data) are propagated to the output layer through the hidden layer, this process is called forward propagation. Then, the output neurons' values (i.e., choice probabilities) are compared with the observed choices to compute the function $J(\mathbf{w})$ described in Equation $(A1)$. The optimisation mechanism is then conducted by propagating J backward to the input layers through the hidden layer. To adjust the weights, the backward propagation process includes taking the partial derivative of the error J with respect to the weights, called the gradient vector $\mathbf{g}$. Along with an adaptive or predefined learning rate value, $\mathbf{w}$ values are re-adjusted.

The process of error (forward and backward) propagation is repeated iteratively until a pre-specified stopping criterion is achieved. This training mechanism is known as back-propagation, and constitutes the most popular approach to train neural networks (Rumelhart et al., 1988). However, it should be noted that moving toward a local minimum is one of the widely reported risks associated with this back-propagation approach (Iyer and Rhinehart, 1999; Park et al., 1996). As such, it is always recommended to train the network more than once to minimise the probability of ending up with a sub-optimal trained network. A pseudocode of the ANN training can be found below, and for comprehensive description of ANNs training interested readers are referred to Bishop (2006).

Appedix 2. Results of Swiss Metro data

This section presents the outcome of applying the proposed approach on the Swiss Metro data (Bierlaire et al., 2001). The data are pre-processed such that only travel time and cost attributes are considered.[17] We used the processed data to train a three-layers fully connected network (see Alwosheel et al. (2018) for ANN complexity adjustment for Swiss Metro data). Note that the training process is similar to the process conducted in section 3 (e.g., same training built-in algorithm, and the k-folds cross-validation method).

Table A1
Performance of the trained ANN

Performance metric	Null model	ANN	Linear-additive RUM
Final Log-likelihood	−9849.2	−6485.9	−6714.54
Cross-entropy	−1.09	−0.70	−0.75
ρ^2	0	0.36	0.32

Table A2 shows, for each alternative (car, Swiss Metro, train), five prototypical examples synthesised using the trained ANN. Note that each example is independently synthesised (i.e., the initial inputs are independently initialised). To facilitate inspection, we employ a so-called vertical heat-map, where high values are depicted red and low values are depicted blue.

[16] Note that in case of training deeper or more complex ANNs, sophisticated algorithms inspired form the gradient descent (e.g., stochastic gradient descent) are used.

[17] The minimum and maximum values of data are normalised to −1 and +1.

Table A2
Synthesised prototypical examples

		TC			TT		
		Car	SM	Train	Car	SM	Train
Train	Ex. 1	94.48	252.83	10.72	110.32	99.07	150.23
	Ex. 2	94.48	252.83	10.72	110.32	99.07	150.23
	Ex. 3	94.48	252.83	10.72	110.32	99.07	150.23
	Ex. 4	94.48	252.83	10.72	110.32	99.07	150.23
	Ex. 5	94.48	252.83	10.72	110.32	99.07	150.23
SM	Ex. 1	93.91	0.56	105.36	170.23	74.78	158.22
	Ex. 2	93.91	0.56	105.36	170.23	74.78	158.22
	Ex. 3	93.91	0.56	105.36	170.23	74.78	158.22
	Ex. 4	93.91	0.56	105.36	170.23	74.78	158.22
	Ex. 5	93.91	0.56	105.36	170.23	74.78	158.22
Car	Ex. 1	77.80	129.12	85.66	129.28	139.11	258.43
	Ex. 2	77.80	129.12	85.66	129.28	139.11	258.43
	Ex. 3	77.80	129.12	85.66	129.28	139.11	258.43
	Ex. 4	77.80	129.12	85.66	129.28	139.11	258.43
	Ex. 5	77.80	129.12	85.66	129.28	139.11	258.43

A number of inferences can be made based on Table A2. First, although each example is independently initialised, the synthesised prototypes for each alternative are almost identical, implying that the ANN has less flexibility due to using only two attributes. Second, the synthesised examples show that the ANN has learned the expected relations. For example, the prototypical examples in which a travel mode is chosen are associated with relatively low travel cost for that mode.

The prototypes shown in Table A2 are cross-validated using the standard linear-additive RUM-MNL (estimation results of RUM-MNL model are shown at Table A3). We use this estimated choice model to inspect whether the synthesised prototypical examples would also be considered prototypical examples from the choice model's perspective. As expected, the estimated choice model returns very high choice probabilities for all prototypes (resulted probabilities are similar to the results reported at Table 6).

Table A3
Estimation results of RUM-MNL model

No. of observations	9036	
Final LL	−6714.54	
ρ^2	0.32	
Attribute	*Est.*	*Rob. t-values*
TT	−2.01	−55.7
TC	−1.03	−24.79

Finally, we use the Swiss Metro data to show how the proposed method can be used to detect or flag a (deliberately) poorly trained ANN. More specifically, we would like to present a situation where the synthesised prototypes show patterns that are unexpected by the analyst, signalling problems with the trained ANN. To do so, we deliberately train a far too complex ANN that consists of two hidden layers (with 500 nodes each). The complexity of this ANN is much higher than the complexity of the problem at hand, which results in a poor model that fails to approximate the underlying data generating process (Vapnik, 2013). For this network, we did not apply the k-folds cross-validation method (the data are randomly separated into two parts: 80% of the data is used for training and the remaining 20% is used for testing). The performance of the trained ANN on training and testing data is presented at Table A4. Results show that the trained ANN obtained excellent prediction performance in the training data but very poor performance in the testing data. This result in itself already signals a so-called overfitting problem, which occurs when the ANN is excessively complex comparing to the underlying data generating process. The excellent performance in the training data is obtained because the ANN complexity allows for fitting the training data perfectly (i.e., including noise artefacts). We choose to apply an overfitting scenario because it is a common mistake due to the ANN capability and flexibility (Abu-Mostafa et al., 2012), although clearly for this extreme overfitting situation, inspection of conventional performance metrics would already suggest to the analyst that something is wrong with the ANN.

Table A4
Performance of the trained ANN (with two hidden layers)

Performance metric	Training data	Testing data
Cross-entropy	−0.14	−2.09
Hit-rate	0.94	0.63

Table A5 shows, for each alternative (car, Swiss metro, train), five prototypical examples synthesised using the trained ANN. Note that each example is independently synthesised (i.e., the initial inputs are independently initialised). To facilitate inspection, we employ a so-called vertical heat-map, where high values are depicted red and low values are depicted blue.

Table A5
Synthesised prototypical examples.

		Travel Cost (CHF)			Travel Time (min)		
		Car	SM	Train	Car	SM	Train
Train	Ex. 1	86.90	1069.96	463.03	145.21	85.86	162.74
	Ex. 2	86.58	16.27	56.07	142.81	101.38	179.01
	Ex. 3	85.85	812.39	377.17	142.95	86.94	176.58
	Ex. 4	81.63	1195.12	387.19	146.57	67.47	246.98
	Ex. 5	85.85	812.40	377.17	142.95	86.94	176.58
SM	Ex. 1	82.70	12.29	9.77	154.68	118.15	194.99
	Ex. 2	103.07	12.30	280.78	143.56	90.65	152.66
	Ex. 3	91.07	12.34	85.23	117.82	98.20	190.21
	Ex. 4	94.09	523.81	399.89	150.27	90.07	170.77
	Ex. 5	94.10	523.81	399.89	150.27	90.07	170.77
Car	Ex. 1	93.44	474.00	359.49	146.01	92.04	173.51
	Ex. 2	93.44	474.00	359.49	146.01	92.04	173.51
	Ex. 3	93.44	474.01	359.49	146.01	92.05	173.51
	Ex. 4	93.44	474.01	359.49	146.01	92.03	173.51
	Ex. 5	67.60	191.72	49.38	149.23	107.10	137.95

The prototypes shown at Table A5 clearly reveal that the ANN has not really learned the expected patterns. For example, considering the first train prototype, the car alternative is actually more attractive than the train alternative (cheaper and faster). This pattern is not expected by travel behaviour analyst. Therefore, the network in this case cannot be trusted.

Appedix 3. Specifications of the linear-additive random utility maximisation model

Table A1 shows the observed utility function for the linear-additive random utility maximisation (RUM) model used in this study (see Table 1 for attributes' name, notation, and description). The model is estimated in Multinomial Logit (MNL) form.

Table A1: Utility function specifications

$$V_{Drive} = ASC_{Drive} + \beta_{TT}TT_{Drive} + \beta_{TC}TC_{Drive} + \beta_{AG\_TC}(AG * TC_{Drive}) + \beta_{DIS\_TC}(DIS * TC_{Drive}) + \beta_{FEM\_TC}(FEM * TC_{Drive}) + \beta_{DL}DL + \beta_{CO}CO + \beta_{TRAF}TRAF$$

$$V_{PubTr} = ASC_{PublTr} + \beta_{TT}TT_{PubTr} + \beta_{TC}TC_{PubTr} + \beta_{AG\_TC}(AG*TC_{PubTr}) + \beta_{DIS_TC}(DIS*TC_{PubTr}) + \beta_{FEM\_TC}(FEM*TC_{PubTr}) + \beta_{BS}BS + \beta_{INTER}INTER$$

$$V_{Walk} = ASC_{Walk} + \beta_{TT}TT_{Walk} + \beta_{TC}TC_{Walk} + \beta_{AG\_TC}(AG * TC_{Walk}) + \beta_{DIS\_TC}(DIS * TC_{Walk}) + \beta_{FEM\_TC}(FEM * TC_{Walk})$$

Notations

V_i	Observed part of utility of alternative i
ASC_i	Specific constant of alternative i
β_{TT}	Taste parameter associated with travel time attribute
β_{TC}	Taste parameter associated with travel cost attribute
β_{AG_TC}	Taste parameter associated with interaction between age and travel cost attribute
β_{DIS_TC}	Taste parameter associated with interaction between travel distance and travel cost attribute
β_{FEM_TC}	Taste parameter associated with interaction between gender and travel cost attribute
β_{DL}	Taste parameter associated with driving license attribute

β_{CO} Taste parameter associated with number of owned car attribute
β_{TRAF} Taste parameter associated with traffic variability attribute
β_{BS} Taste parameter associated with bus scale attribute
β_{INTER} Taste parameter associated with number of interchanges attribute

References

Abu-Mostafa, Y.S., Magdon-Ismail, M., Lin, H.-T., 2012. Learning from Data, vol. 4. AMLBook, New York, NY, USA.

Alwosheel, A., van Cranenburgh, S., Chorus, C.G., 2018. Is your dataset big enough? Sample size requirements when using artificial neural networks for discrete choice analysis. J. Choice Model. 28, 167–182. http://dx.doi.org/10.1016/j.jocm.2018.07.002.

Batista, G.E., Prati, R.C., Monard, M.C., 2004. A study of the behavior of several methods for balancing machine learning training data. ACM SIGKDD explorations newsletter, 6 (1), 20–29.

Bierlaire, M., Axhausen, K., Abay, G., 2001. The acceptance of modal innovation: the case of Swissmetro. In: Paper Presented at the Swiss Transport Research Conference.

Bishop, C.M., 1995. Neural Networks for Pattern Recognition. Oxford university press.

Bishop, C.M., 2006. Pattern Recognition and Machine Learning. springer.

Breiman, L., 2001. Statistical modeling: the two cultures (with comments and a rejoinder by the author). Stat. Sci. 16 (3), 199–231.

Chawla, N.V., Bowyer, K.W., Hall, L.O., Kegelmeyer, W.P., 2002. SMOTE: synthetic minority over-sampling technique. J. Artif. Intell. Res. 16, 321–357.

Chiang, W.-y. K., Zhang, D., Zhou, L., 2006. Predicting and explaining patronage behavior toward web and traditional stores using neural networks: a comparative analysis with logistic regression. Decis. Support Syst. 41 (2), 514–531.

Chollet, F., 2015. Keras.

Erhan, D., Bengio, Y., Courville, A., Vincent, P., 2009. Visualizing higher-layer features of a deep network. University of Montreal, p. 1, 1341(3).

Fernández, C., Fernández, A., 2019. Ethical and legal implications of AI recruiting software. ERCIM NEWS 116, 22–23.

Garson, G.D., 1991. Interpreting neural-network connection weights. AI Expet. 6 (4), 46–51.

Golshani, N., Shabanpour, R., Mahmoudifard, S.M., Derrible, S., Mohammadian, A., 2018. Modeling travel mode and timing decisions: comparison of artificial neural networks and copula-based joint model. Travel Behaviour and Society 10, 21–32. http://dx.doi.org/10.1016/j.tbs.2017.09.003.

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., et al., 2014. Generative adversarial nets. In: Paper Presented at the Advances in Neural Information Processing Systems.

Hagenauer, J., Helbich, M., 2017. A comparative study of machine learning classifiers for modeling travel mode choice. Expert Syst. Appl. 78, 273–282. http://dx.doi.org/10.1016/j.eswa.2017.01.057.

Haykin, S.S., 2009. Neural Networks and Learning Machines, vol. 3. Pearson Upper, Saddle River, NJ, USA.

He, H., Garcia, E.A., 2008. Learning from imbalanced data. IEEE Trans. Knowl. Data Eng. (9), 1263–1284.

Hillel, T., Elshafie, M.Z., Jin, Y., 2018. Recreating passenger mode choice-sets for transport simulation. Proc. Instit, Civil Eng.-Smart Infrastruct. Construct. 1–49.

Iyer, M.S., Rhinehart, R.R., 1999. A method to determine the required number of neural-network training repetitions. IEEE Trans. Neural Netw. 10 (2), 427–432.

Karlaftis, M.G., Vlahogianni, E.I., 2011. Statistical methods versus neural networks in transportation research: differences, similarities and some insights. Transp. Res. C Emerg. Technol. 19 (3), 387–399.

Kingma, D.P., Ba, J., 2014. Adam: A Method for Stochastic Optimization. 1412.6980.

Krizhevsky, A., Sutskever, I., Hinton, G.E., 2012. Imagenet classification with deep convolutional neural networks. In: Paper Presented at the Advances in Neural Information Processing Systems.

Kulesza, T., Burnett, M., Wong, W.-K., Stumpf, S., 2015. Principles of explanatory debugging to personalize interactive machine learning. In: Paper Presented at the Proceedings of the 20th International Conference on Intelligent User Interfaces.

Lee, D., Derrible, S., Pereira, F.C., 2018. Comparison of four types of artificial neural network and a multinomial Logit model for travel mode choice modeling. Transp. Res. Rec. http://dx.doi.org/10.1177/0361198118796971, 0361198118796971.

Lipton, Z.C., 2016. The Mythos of Model Interpretability arXiv preprint arXiv:1606.03490.

McFadden, D., 1973. Conditional Logit Analysis of Qualitative Choice Behavior.

Miller, T., 2017. Explanation in Artificial Intelligence: Insights from the Social Sciences arXiv preprint arXiv:1706.07269.

Montavon, G., Samek, W., Müller, K.-R., 2018. Methods for interpreting and understanding deep neural networks. Digital Signal Processing 73. http://dx.doi.org/10.1016/j.dsp.2017.10.011.

Mordvintsev, A., Olah, C., Tyka, M., 2015. Inceptionism: going deeper into neural networks. Google Research Blog, p. 5. Retrieved June, 20(14).

Nguyen, A., Dosovitskiy, A., Yosinski, J., Brox, T., Clune, J., 2016. Synthesizing the preferred inputs for neurons in neural networks via deep generator networks. In: Paper Presented at the Advances in Neural Information Processing Systems.

Nguyen, A., Yosinski, J., Clune, J., 2015. Deep neural networks are easily fooled: high confidence predictions for unrecognizable images. In: Paper Presented at the Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition.

Nguyen, A., Yosinski, J., Clune, J., 2016. Multifaceted feature visualization: uncovering the different types of features learned by each neuron in deep neural networks arXiv preprint arXiv:1602.03616.

Park, Y.R., Murray, T.J., Chen, C., 1996. Predicting sun spots using a layered perceptron neural network. IEEE Trans. Neural Netw. 7 (2), 501–505.

Ribeiro, M.T., Singh, S., Guestrin, C., 2016. "Why Should I Trust You?": Explaining the Predictions of Any Classifier. Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining (pp. 1135-1144). http://dx.doi.org/10.1145/2939672.2939778.

Rumelhart, D.E., Hinton, G.E., Williams, R.J., 1988. Learning representations by back-propagating errors. Cognitive modeling 5 (3), 1.

Samek, W., Wiegand, T., Müller, K.-R., 2017. Explainable Artificial Intelligence: Understanding, Visualizing and Interpreting Deep Learning Models arXiv preprint arXiv:1708.08296.

Shmueli, G., 2010. To explain or to predict? Stat. Sci. 25 (3), 289–310.

Sifringer, B., Lurkin, V., Alahi, A., 2018. Enhancing discrete choice models with neural networks. In: Paper Presented at the hEART 2018–7th Symposium of the European Association for Research in Transportation Conference.

Simonyan, K., Vedaldi, A., Zisserman, A., 2013. Deep inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps arXiv preprint arXiv:1312.6034.

Van Cranenburgh, S., Alwosheel, A., 2019. An artificial neural network based approach to investigate travellers' decision rules. Transp. Res. C Emerg. Technol. 98, 152–166. http://dx.doi.org/10.1016/j.trc.2018.11.014.

Vapnik, V., 2013. The Nature of Statistical Learning Theory. Springer science & business media.

Yosinski, J., Clune, J., Nguyen, A., Fuchs, T., Lipson, H., 2015. Understanding neural networks through deep visualization arXiv preprint arXiv:1506.06579.