ERASMUS UNIVERSITY ROTTERDAM

Using machine learning on feature selection

A technical analysis on combining features and their knockoffs within a recurrent neural network

Master's Thesis

Author
Raymond A. Biharie

Supervisor
dr. Andreas Pick

Second assessor
dr. Yutao Sun

A thesis submitted in partial fulfillment of the requirements for the Master degree of Econometrics and Management Science

at
Erasmus School of Economics
Department of Econometrics

April 2020

Declaration of Authorship

I, Raymond A. Biharie, declare that this thesis titled "Using machine learning on feature selection: A technical analysis on combining features and their knockoffs within a recurrent neural network" and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at this University.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.
- Where I have copyright of this thesis, the intellectual copyright of contributions made by the thesis supervisor, which may include important research ideas and data, are also acknowledged.
- Electronic versions of the thesis are directly available for inclusion in any EUR thesis database and repository such as the Master Thesis Repository of the Erasmus University Rotterdam and may be consulted upon request. Clear agreements about issues such as confidentiality are made.

Signed: 

"No matter how life may turn out,
always remember that
you did well"

R.A. Biharie

ERASMUS UNIVERSITY ROTTERDAM

Abstract

Erasmus School of Economics
Department of Econometrics

Master of Science

by Raymond A. Biharie

Feature selection, or variable selection, is an important task that is needed within a lot of disciplines. This study applies deep learning on the selection of features using a proposed model. This study contributes to existing literature by examining the performance of a recurrent neural network in a model where both knockoffs of features, or almost identical copies of variables, and the true features are combined within one model framework. This model is used for variable selection purposes where a controlled false discovery rate is used. A simulation study shows that adding additional information with a recurrent neural network does not seem to improve a selection of a subset of features and that combining both knockoff features and true features does only give an advantage to subset selection in case of a feedforward neural network.

Keywords: *Neural network, feedforward neural network, recurrent neural network, Elman network, machine learning, deep learning, DeepPINK, model-X knocknoffs, false discovery rate, feature selection, simulation.*

J.E.L. classification: *C01, C15, C45.*

Acknowledgements

After months of hard work and dedication I can finish my career as a graduate student Econometrics and Management Science at Erasmus University Rotterdam. Throughout the years I have had great times, but I have also experienced some hardships and have learned that life is not always colourful. In my years as a student, I have met great people who stood by my side and who gave me advice which also enriched my experiences in life and made my life as a student easier. Therefore, I want to use this part to express my gratitude to these people.

First of all, I want to thank my supervisor dr. Andreas Pick for guiding me during my writing process. I want to thank him for his patience during my writing process as I needed to narrow down my research to a more specific problem than I had initially proposed. I also want to thank dr. Yutao Sun for the intellectual discussions and critical comments about my thesis, which have provided me with some valuable insights. I am thankful to Erasmus University Rotterdam and all the people I came in contact with who gave me the necessary skills and knowledge to work on my thesis and to finish my curriculum at the Master's program.

Secondly, I want to thank Mohammed Oemar and his family who gave me the kindness and support that I needed during my studies. Their presence and the conversations I have had with them throughout the years about life were an enlightenment and made me feel less lonely than I would otherwise feel at the Master's program. I have also learned a lot from him and from some other people about all the upsides and downsides of living in this society. About the difference between people who make strategic choices in life and about people who are selfless, about greed and about contentment, about giving socially acceptable answers and about giving your true opinion, about acting like having a positive lifestyle that can be measured up to the standards of society and about staying and acting true to yourself. I am thankful for those life lessons that I have had in the last 7 years.

I want to thank Anna-Marie Loeff and Hildemarij Fortanier-Mesker who I have met 7 years ago. They were really kind to me and gave me a chance to work with them. I am truly grateful for the experiences I have had, for the work they were doing and for their kind hearts and selfless acts, which have helped me into forming my personality. Moreover, I want to thank Hamza Aammari, Haldun Simsek and Farid Aissati for accepting me into their group of friends 10 years ago when I was alone. I also want to thank Patricia Jansen for her lessons on psychology and communication. Her lessons have changed my way of thinking and have made a positive impact on how I look at different kinds of situations.

Thirdly, I want to say my thanks to all the people who provided me with the basic tools and care that I needed to reach this point in life. I want to say thanks to my parents, my family, people I have met in life and friends of my brother who are always considerate to ask about me, in particular Askin Sarikaya and a friend of his. I want to thank my dad for opening my eyes and for trying to learn me bad norms and values, which actually showed me what not to do and how not look at certain aspects in life. I also want to thank all of the people who never believed in me, back then and now, and who always treated me like I was less important, which actually gave me the strength and the motivation to get more out my educational career than I would

otherwise have gotten. I want to thank all the people who were always true and sincere to me and to my brother.

I want to thank Mohammad Nazir who has always been there for me ever since I have met him 10 years ago. Since I have met him, he and my brother have made a significant impact on my personality. All the deep and philosophical conversations I have had so far with him and my brother about life have sculptured my thoughts and have hopefully made me into a better person than I was before meeting him.

Most importantly, I want to express my sincere gratitude to my brother, Armand Biharie, who, up until this day, has always stood by my side, never stopped believing in me and who has done too much already for me and for others in life. I hope that I am able to return all his favours one day in life and I hope that finishing my career as a student is one step towards reaching that goal.

Lastly, I want to add that life is too short for it to be led by ego, greed, jealousy and standards of society, which I am also influenced by as I have my flaws. For all the people mentioned in this part I truly hope that these people all get there in life where they want to be and that all of them find happiness, contentment, always stay healthy and will become a better person everyday with better norms and values. I sincerely hope all the best for these people and for their families. Thank you, all of you.

Contents

List of Tables

Abbreviations

DeepPINK	**Deep** feature selection using **P**aired-**I**nput **N**onlinear **K**nockoffs
DGP	**d**ata **g**enerating **p**rocess
EDP	**E**lman neural network with **D**eep**P**INK
ENN	**E**lman **n**eural **n**etwork
FDP	**F**alse **D**iscovery **p**roportion of Type I errors
FDR	**F**alse **D**iscovery **R**ate
FNN	**f**eedforward **n**eural **n**etwork
i.i.d.	**i**ndependent and **i**dentically **d**istributed
MAPE	**m**ean **a**bsolute **p**rediction **e**rror
MLP	**M**ulti**l**ayer **p**erceptron
MSE	**m**ean **s**quared **e**rror
MSPE	**m**ean **s**quared **p**rediction **e**rror
NN	**n**eural **n**etwork
OLS	**O**rdinary **L**east **S**quares
RNN	**r**ecurrent **n**eural **n**etwork

Dedicated to

my mom
for her endless love

1 Introduction

Modeling and feature selection (variable selection) take a prominent role in the assessment of econometric models. Companies and regulating entities use feature selection to get more accurate models that are used for decision-making. Feature selection can lead to subset selection and model simplification. Feature selection can be done using parametric statistical tools and non-parametric statistical tools. One non-parametric variant is the use of a neural network algorithm (Arnerić, Šestanović, & Aljinović, 2014). Neural networks (NNs) are networks consisting of neurons from the nervous system and the use of neural network models has been extended over the years to several fields, like e.g. astronomy, bionics, Information Technology, economics and econometrics. Neural networks are a set of machine learning methods, where machine learning can be seen as a subfield of artificial intelligence.

NN models are used for non-linear models and are applicable for forecasting purposes (e.g. Franses and Draisma (1997)), classification, recognition purposes and regressions (Cheng & Titterington, 1994; Gómez-Ramos & Venegas-Martínez, 2013). NN models are no estimation models, which use Maximum Likelihood estimation, but which make use of the backpropagation algorithm (Rumelhart, Hinton, & Williams, 1986). This backpropagation algorithm can be seen as a gradient method (Gómez-Ramos & Venegas-Martínez, 2013) when a feedforward neural network with multiple hidden layers, also known as a Multilayer Perceptron (MLP) (Arnerić et al., 2014), is used. The advantage of using NNs, without having to depend on prior information, to select relevant features of a dataset is a favourable aspect (Chong, Han, & Park, 2017).

NNs form a graphical representation of nodes from a process that flows from input to output with in-between processes that influence the input nodes until the output is reached. Architectures of NNs depend on the interconnectedness between the nodes, the configuration of the nodes and the nature of the operations at each node reliable on the problem at hand (Cook & Smalter Hall, 2017). NNs are build upon the concept of a perceptron, which has three layers. The input layer directs the input into the second computational layer (the hidden layer), which processes the inputs. The output layer is the third layer with nodes that contain the output (Gradojevic & Yang, 2000). Activation functions can be used to determine which neurons should be turned on or off inside a hidden layer (Kuan, 2006). Activation functions are not necessary to build a working NN (Kuan, 2006), but are used such that the network does not fold together into a linear model and such that an NN is one of the possible choices for modeling the data.

The feedforward neural network (FNN) is considered to be the basis for other adaptations of NNs. An extension on the FNN is the recurrent neural network (RNN), which is a network which has a memory such that nodes from a previous step in the network will be taken into account in the following steps (Elman, 1990), which gives an advantage of the RNN compared to FNNs to create long temporal structures (Sundermeyer, Alkhouli, Wuebker, & Ney, 2014). In other words: The RNN is a dynamic form of a neural network algorithm in the way the information

of the hidden layer from the previous step is incorporated into the model again for determining the output (Cook & Smalter Hall, 2017).

There are several variants of the RNN model, including, among others: Real-Time Recurrent Learning (Sutskever, 2013; Williams & Zipser, 1989), Time-Delay Neural Network (Waibel, Hanazawa, Hinton, Shikano, & Lang, 1989), Long-Short Term Memory (Hochreiter & Schmidhuber, 1997), Echo-State Network (Jaeger & Haas, 2004), Elman Network (Elman, 1990; Kuan & Liu, 1995; Kuan, 2006; Sitte & Sitte, 2002) and Jordan Neural Network (Jordan, 1986, 1997; Kuan, 2006). This paper will focus on the Elman neural network. In contrast to the Jordan network, where the updated output is again used in the next step when updating the weights, the Elman neural network (ENN) looks more at the internal neural responses and takes the hidden layer nodes from the previous step into account in the hidden layer in the next step during the training process (Kuan, 2006).

Given that in reality, not all information may be relevant for explaining the output (Foygel Barber & J. Candès, 2015; Cook & Smalter Hall, 2017; Jianming, 1998; Gao & Jojic, 2016; Rohwer, 1990; Gradojevic & Yang, 2000), this paper tries to complement the problem of filtering out relevant features/variables by using the model-X knockoffs framework in combination with a DeepPINK (Deep feature selection using Paired-Input Nonlinear Knockoffs) model framework introduced by (Candès, Fan, Janson, & Lv, 2018). The *model-X knockoffs* framework is in essence used to create copies which are almost identical to the original X-variables of a dataset, whereas *DeepPINK* is used to combine these copies in combination with the original variables in order to verify if this method may be beneficial for selecting relevant features in a dataset. The mathematical derivation of the model-X knockoffs and DeepPINK framework will be outlined in section 2. The model-X knockoffs framework will be combined with an ENN. In other words:

This study will try to accommodate a solution to the problem of feature selection for an RNN by using a model-X knockoffs framework in combination with a DeepPINK model framework, where the objective of this study lies in examining to what extent the algorithm can be improved in terms of feature selection by employing these frameworks into an RNN.

This way the proposed method may lead to dimension reduction by selecting the relevant features (Herbrich, Keilbach, Graepel, Bollmann-sdorra, & Obermayer, 1998) while using the memory property of an RNN. Gençay and Liu (1997) compare the relative prediction performance of the ENN (Elman, 1990) with the FNN. They find that the ENN predicts with a smaller noise than the FNN. This study will test if the error for selecting features will be smaller when applying a combination of the model-X knockoffs framework and the DeepPINK model framework for an RNN.

A knockoff filter will be implemented in order to get model-X knockoff features. The knockoff filter was first introduced by Foygel Barber and J. Candès (2014, 2015) as variable selection method to control the FDR (the false discovery rate or the expected number of false discoveries relative to all discoveries) in a statistical linear model where the number of observations is at least as high as the number of variables. An advantage of using the knockoffs framework is that the framework can be used in combination with several test statistics. In the work of

Foygel Barber and J. Candès (2016) they use the knockoffs framework in a two-stage model which was used for FDR control in high-dimensional linear models, where the number of variables is higher than the number of observations. Katsevich and Sabattiy (2018) embedded the knockoff filter to a multilayer knockoff filter (MKF), where they combine the multilayer knockoffs framework of Foygel Barber and J. Candès (2015) and the framework for multilayer FDR control of Foygel Barber and Ramdas (2016). This way they try to have FDR control for a large number of variables where they take the possibility of grouping of important variables into account. Foygel Barber, J. Candès, and J. Samworth (2019) look at the model-X knockoffs framework where the distribution of a subset with p features is not known for sure and needs to be estimated. The p features are used to derive p knockoff copies of the original features in order to check if there are not too many irrelevant features chosen to explain the dependent outcome (Foygel Barber et al., 2019). Romano, Sesia, and J. Candès (2018) use deep-generative models for sampling approximate knockoffs in case the distribution is not specified. A Metropolis-Hastings formulation is used in the work of Bates, Candès, Janson, and Wang (2019) for sampling knockoff variables. Sesia, Katsevich, Bates, Candès, and Sabatti (2019) develop a method that is used to localize causal genetic variants and to control the FDR with knockoff copies for the genotypes as controls. In this study the model-X knockoffs will be created as was done in the study of Candès et al. (2018) and Lu, Fan, Lv, and Stafford Noble (2018).

The objective of this study is to develop a machine learning technique for combining the model-X knockoffs framework with the DeepPINK model framework into an ENN, where a supervised learning method will be applied. Supervised learning methods do require to have knowledge of the underlying modeling process. This means that the model will map given input to known output. The accuracy in feature selection within the proposed model will be assessed by looking at error rates. Up until now, research using the DeepPINK model framework has only focused on the filtering of input variables in combination with an FNN (Lu et al., 2018). The extension of the existing literature in this study is relevant for practical purposes as well as for scientific purposes, which will be interesting for data scientists and data analysts who work with NNs. This study will give an answer to the following questions:

- Will an ENN algorithm combined with the model-X knockoffs framework and the DeepPINK model framework prove to be better in feature selection than an ENN where only the knockoffs are used as variables?
- What differences in feature selection arise when considering an FNN with model-X knockoffs and DeepPINK?
- Will varying the initial settings alter conclusions regarding feature selection, where the number of hidden layers, number of observations and number of features (variables) will be varied?

A simulation study will be developed to assess how well the proposed model performs in selecting features compared to other NNs when subjected to several forms of model-X knockoffs and several levels of non-zero signals for the parameters. The outcomes of all examined models will be used to assess the reliability with the mean squared error and mean squared prediction error. An

empirical application will be done using the proposed model to see how the model performs on real data.

The remainder of this study is structured as follows: Section 2 will discuss the proposed model and the methodology behind this model. Techniques for measuring the performance and accuracy of the proposed model will also be discussed. In section 3 the results with the use of the proposed model will be given where a simulation study will be implemented in order to assess the reliability of the outcomes from the model. Subsequently, I will give a brief outline of the used data and an empirical analysis will be performed on the data to examine how well the model performs in practice. Section 4 will conclude the research after which some recommendations for further research will be given.

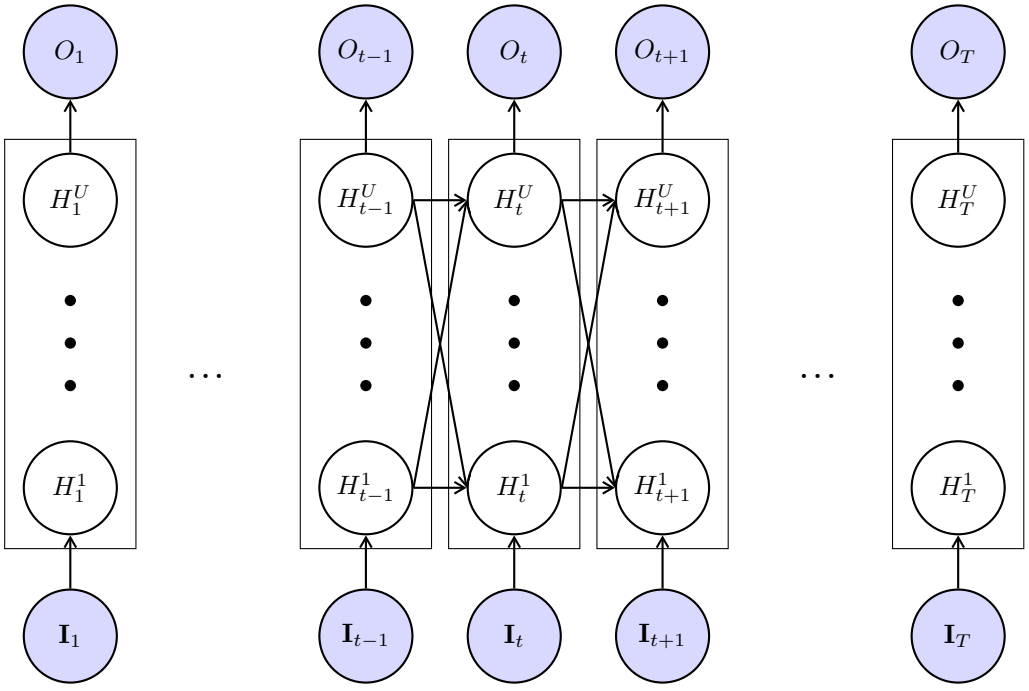
2 Methodology

Before outlining the methodology that is used in this paper, I will start with defining the ENN and then I will follow with the extension of the ENN using a combination of the model-X knockoffs framework and the DeepPINK framework. Comparison tests and measures for accuracy and performance are discussed.

2.1 Recurrent neural networks

For simplicity, I will assume that the set of nodes in the proposed model is finite, where the set of nodes can also be infinite (de Paula, 2016). Note that a link between nodes can also be self-referential, such that a node links to itself, and that two nodes can have a reciprocal link, such that the relationship between the nodes goes both ways (de Paula, 2016). For this paper I assume that the network is directed (only a link from one node to the other exists and not the other way around) and weighted. Hence, I do not assume that there are also non-existent links within the network (Chandrasekhar, 2016). The configuration of the model that is used in this paper, regarding the number of hidden layers, number of nodes and activation function is given in Appendix A.

The fully connected ENN with U hidden layers, one input variable and one output variable can graphically be depicted as follows (Sutskever, 2013):



Here the $N \times 1$ vector I_t is the input at step t in the training process for $t = 1, ..., T$ updates for one variable with N observations, the $N \times 1$ vector H_t^u is the u^{th} hidden layer node at step t for $u = 1, ..., U$ hidden layers and the $N \times 1$ vector O_t is the output neuron at step t. This architecture is fully connected as all hidden layer nodes have context units (Lobbrecht, Dibike, & Solomatine, 2005), which also means that the number of weights and context units depend on the number of nodes in the network and in the hidden layer, respectively. One should keep in mind that the convergence of the weights is done at a faster rate when there are fewer neurons compared to the number of data points (Herbrich et al., 1998).

The ENN with 1 hidden layer could be defined as follows for the t-th step (Binner, Elger, Nilsson, & Tepper, 2005; Chappelier, Gori, & Grumbach, 2001):

$$\begin{aligned} \nu_t &= w_{1,t-1} I_t + b_1 + H_{t-1} \\ H_t &= f(\nu_t) \\ O_t &= w_{U+1,t-1} H_t \end{aligned} \tag{2.1}$$

Here $f(\cdot)$ is a non-linear activation function. For the first run, when $t = 1$, the weights with subscript $t - 1$ are the initialized weights before any update has been done in the training process. The context unit H_{t-1} is not yet present during the first step in the training process. That is, the weight of the context unit is set to 0 instead of 1 during the first step of the training process before any update of the weights (Elman, 1990). This representation is one of a recurrent network, given that it is dynamic in the way that hidden layer nodes for the previous step in the training process are used to determine the hidden layer nodes at the current step. The context unit does affect all nodes within a hidden layer in the proposed model in this study, in line with (Elman, 1990). The ReLU function for one node in the hidden layer at update step t is defined as:

$$\begin{aligned} f(\nu_t) &= max(0, \nu_t) \\ f'(\nu_t) &= \begin{cases} 1, & \text{if } \nu_t > 0 \\ 0, & \text{if } \nu_t \leq 0 \end{cases} \quad \text{for } t = 1, ..., T, \end{aligned} \tag{2.2}$$

where each value within the $N \times 1$ vector $f'(\nu_t)$ is set to 1 or 0 according to the threshold. The derivative of the ReLU activation function is used to get the change in slope in order to see the relevance of the change in the weight (Chong et al., 2017; Sun, 1999).

The formulation of the ReLU function shows that the outcome of the activation function determines which of the nodes from the hidden layer(s) are important, where the size of importance is seen by the weight when a hidden layer node is set to 1 and where the node is not important if the node is set to 0, regardless of the weight corresponding to the node.

2.2 Model-X knockoffs and the false discovery rate

This paper will use the false discovery rate (FDR) to calculate threshold values in order to filter out irrelevant variables. Benjamini and Hochberg (1995) state that the number of falsely rejected null hypotheses are inversely related to the seriousness of the loss that arises from falsely rejected hypotheses. To this end, they develop a measure to look at the expected proportion of errors due to falsely rejected hypotheses (call this E(FDP)), which is introduced as the FDR. The FDR is a measure to control for Type I errors (Benjamini & Hochberg, 1995), where Type I errors are known as the errors that occur when falsely rejecting hypotheses, while these are actually true. In the study of Benjamini and Hochberg (1995) the FDR is formulated as the number of type I errors (call this V) proportional to the sum of (1) the number of type I errors and (2) the number of justified rejected hypotheses (call the latter S):

$$FDR := E(FDP) = E\left[\frac{V}{(V+S)}\right], \tag{2.3}$$

where he assumption is that $\frac{0}{0} = 0$ (Lu et al., 2018).

The model in this paper will be used to do feature selection. To this end, a dataset needs to be taken into account. Assume that there exists a dataset $\{\mathbf{X}, \mathbf{y}\}$ with $k+1$ independent identically distributed (i.i.d.) random variables and N observations, where $\mathbf{X} \in \mathbb{R}^{N \times k}$ is a feature matrix and $\mathbf{y} \in \mathbb{R}^N$ is the response vector. The purpose of the model in this paper is to find those $\mathbf{X}$ variables/features which can capture the outcomes in $\mathbf{y}$ the most. The restriction $N > k$ will be put on the dataset, given that it is desired that there will be as less identification problems in estimating the parameters as possible (Foygel Barber & J. Candès, 2015). The model in this paper can be seen as an ordinary least squares (OLS) problem as an NN also aims to minimize the estimation errors.

Most methods for FDR control cannot be used in the neural network framework as these methods use p-values to investigate the Type I errors and as meaningful p-values cannot be obtained (Lu et al., 2018). Lu et al. (2018) try to use the model-X knockoffs framework as an alternative to using p-values, which can then be used as control in the selection of variables by comparing the feature importance between the true and the copied variables. $\mathbf{y}$ is assumed to depend on a subset with $p < k$ features of $\mathbf{X}$. In line with Candès et al. (2018), say that there is an $N \times p$ matrix $\mathbf{X}^S = (\mathbf{x}_1, ...\mathbf{x}_p)^T \subset \mathbf{X}$ with $p < k$ features which have the model-X knockoffs $\tilde{\mathbf{X}}^S = (\tilde{\mathbf{x}}_1, ...\tilde{\mathbf{x}}_p)^T$.

To get FDR control in a neural network setting, one can look at the model-X knockoffs framework (Lu et al., 2018; Candès et al., 2018). The knockoffs framework is used to make copied features which can then function in this study as input in a neural network framework combined with the true features. The knockoff copies are made such that the correlation structure of the copies resemble the correlation structure of the true variables in order to minimize the FDR (Foygel Barber & J. Candès, 2014, 2015). Candès et al. (2018) allocate two properties to the model-X knockoffs:

1. For any subset with p variables (call this subset A), it holds that $(\mathbf{X}^S, \tilde{\mathbf{X}}^S)_{swap(A)} \overset{\text{d}}{=} (\mathbf{X}^S, \tilde{\mathbf{X}}^S)$, such that $\forall i \in A$ the swapped $\mathbf{x}_i$ and $\tilde{\mathbf{x}}_i$ have the same distribution (denoted by $\overset{\text{d}}{=}$) as in the case where these are not swapped.

2. $\tilde{\mathbf{X}}^S \perp \mathbf{y}|\mathbf{X}^S$, such that the knockoff-X $\tilde{\mathbf{X}}^S$ is independent of the outcome $\mathbf{y}|\mathbf{X}^S$.

By comparing the model-X knockoffs with the original features at some target level q, FDR control can be achieved (Lu et al., 2018; Candès et al., 2018). The FDR is controlled at target level q = {0,1}. The target level q means that the FDR is to be at most q irrespective of the parameters in the model (Foygel Barber & J. Candès, 2015). The relevant knockoffs are found by using knockoff statistics $l_i = g_i(d_i, \tilde{d}_i)\ \forall i \in \{1, ..., p\}$ (Candès et al., 2018), which are used as a measure for selection via a threshold to select which features are assumed to be relevant (Foygel Barber & J. Candès, 2015). Here d_i and $\tilde{d}_i$ are respectively scalar measures for the importance of the features of $\mathbf{x}_i$ and $\tilde{\mathbf{x}}_i$ and $g(\cdot, \cdot)$ is an anti-symmetric function. Lu et al. (2018) sort the absolute scalar knockoff statistics $|l_i| > 0$ in descending order and use a threshold value to determine if the calculated knockoff statistics are higher than a user-specified target FDR level $q \in (0, 1)$. That is, the pairwise couples ($\mathbf{x}_i$, $\tilde{\mathbf{x}}_i$) will be selected with the help of the knockoff statistics, using a threshold based on the desired FDR level, to filter the most important features at a certain level. The knockoff test statistic is used to check if the threshold is exceeded. If a knockoff test statistic occurs several times, I start with taking the first until the last knockoff variable having that particular test statistic. As threshold values, two measures can be taken (Lu et al., 2018), which are a threshold for controlling the modified filter (equation (2.4)) of the FDR and one for the exact FDR (equation (2.5)):

$$\theta = min\left\{\alpha \in \mathcal{D}, \frac{|\{i : l_i \leq -\alpha\}|}{max(1, |\{i : l_i \geq \alpha\}|)} \leq q\right\}, \tag{2.4}$$

$$\theta_+ = min\left\{\alpha \in \mathcal{D}, 1 + \frac{|\{i : l_i \leq -\alpha\}|}{max(1, |\{i : l_i \geq \alpha\}|)} \leq q\right\}, \tag{2.5}$$

where $\mathcal{D} = |l_i|$ for $i = \{1, .., p\}$, α is the set of sorted absolute knockoff statistics in descending order and where $|\cdot|$ is the absolute number of times the expression inside the curly brackets holds. If the ratio inside the curly brackets gets larger than the specified target FDR q, then the corresponding variable i is selected as a relevant variable.

Either all variables are disposed ($p = 0$) or chosen ($p = k$) as the p important features. α is equal to zero when the target FDR is set to $q = 1$ and infinite when FDR = 0, which means that the threshold is determined by the setting of the FDR level. This means that there is no threshold when the target FDR level is set to 1 (=100% of the variables are found to have a significant effect), such that all knockoff copies are selected as relevant and that, when the target FDR is 0% ($q = 0$), no knockoff copies are selected as q is always lower than the calculated fraction in equation (2.4) and equation (2.5). In this study I have based the algorithm, for making the thresholds, on the algorithm that is used in the knockoff package in R Core Team (2019), where this package is developed by and used in the study of Candès et al. (2018).

2.3 Model framework

This paper will combine the model-X knockoffs framework with the ENN. Using a subset of p model-X knockoffs allows for selecting features with a controlled FDR. To create the knockoffs the data is assumed to follow a multivariate Gaussian distribution, such that approximate model-X Gaussian knockoffs are used as in the study of Candès et al. (2018) and Lu et al. (2018). This means that $\mathbf{X}^S \sim \mathcal{N}(0, \mathbf{\Sigma})$ with variance-covariance matrix $\mathbf{\Sigma}^{p\times p}$. A characteristic of the variance-covariance matrix in a multivariate normal distribution is that the matrix is positive definite. This means that all elements in the variance-covariance matrix are above 0, such that $\mathbf{\Sigma}^{-1}$ exists. The feature matrix (the matrix with the original variables) will be normalized such that each diagonal element of $\mathbf{\Sigma} = (\mathbf{X}^S)^T\mathbf{X}^S$, taken as $\Sigma_{ii} = ||\mathbf{x}_i||^2 = 1 \ \forall i \in \{1, ..., p\}$ (Foygel Barber & J. Candès, 2015). The assumption is that $\mathbf{\Sigma} = (\tilde{\mathbf{X}}^S)^T\tilde{\mathbf{X}}^S$ and $(\mathbf{X}^S)^T\tilde{\mathbf{X}}^S = \mathbf{\Sigma} - diag(\mathbf{s})$ (Foygel Barber & J. Candès, 2014). This leads to knockoffs, with a correlation structure comparable to the correlation structure of the true features (Lu et al., 2018; Foygel Barber & J. Candès, 2015), $\tilde{\mathbf{X}}^S = \mathbf{X}^S(\mathbf{I} - diag(\mathbf{s})\mathbf{\Sigma}^{-1}) - \tilde{\mathbf{V}}\mathbf{D}$, with $\tilde{\mathbf{V}}$ an orthonormal $N \times p$ matrix (which also means that the columns in the matrix are orthogonal) and with $\mathbf{D}^T\mathbf{D} = 2diag(\mathbf{s}) - diag(\mathbf{s})\mathbf{\Sigma}^{-1}diag(\mathbf{s})$ a Cholevsky decomposition such that:

$$\tilde{\mathbf{X}}^S|\mathbf{X}^S \sim \mathcal{N}(\mathbf{X}^S - diag(\mathbf{s})\mathbf{\Sigma}^{-1}\mathbf{X}^S,\ 2diag(\mathbf{s}) - diag(\mathbf{s})\mathbf{\Sigma}^{-1}diag(\mathbf{s})), \tag{2.6}$$

where $diag(\mathbf{s})$ is a diagonal matrix made with a $p \times 1$ vector $\mathbf{s}$ with all positive elements with the additional condition that the conditional variance-covariance matrix stays positive definite.

The joint distribution of the true features and the knockoffs can then be denoted as (Lu et al., 2018):

$$(\mathbf{X}^S, \tilde{\mathbf{X}})^S \sim \mathcal{N}\left(\begin{pmatrix}0\\0\end{pmatrix}, \begin{pmatrix}\mathbf{\Sigma} & \mathbf{\Sigma} - diag(\mathbf{s})\\ \mathbf{\Sigma} - diag(\mathbf{s}) & \mathbf{\Sigma}\end{pmatrix}\right). \tag{2.7}$$

The value of $\mathbf{s}$ needs to be as large as possible in order to prevent a too large resemblance between the true feature $\mathbf{x}_i$ and the corresponding knockoff $\tilde{\mathbf{x}}_\mathbf{i}$. This leads to a larger statistical power of the used method in this paper (Lu et al., 2018; Foygel Barber & J. Candès, 2015).

The covariance in the distribution of $\tilde{\mathbf{X}}^S|\mathbf{X}^S$ is based on the derivation of the Schur complement (Foygel Barber & J. Candès, 2014) to ensure that the covariance matrix is invertible. If $\mathbf{\Sigma}$ is invertible, it should hold that for matrix $\mathbf{A} = \begin{pmatrix}\mathbf{\Sigma} & \mathbf{\Sigma} - diag(\mathbf{s})\\ \mathbf{\Sigma} - diag(\mathbf{s}) & \mathbf{\Sigma}\end{pmatrix}$ the Schur complement of $\mathbf{\Sigma}$ in $\mathbf{A}$ is derived as (Carlson, 1986):

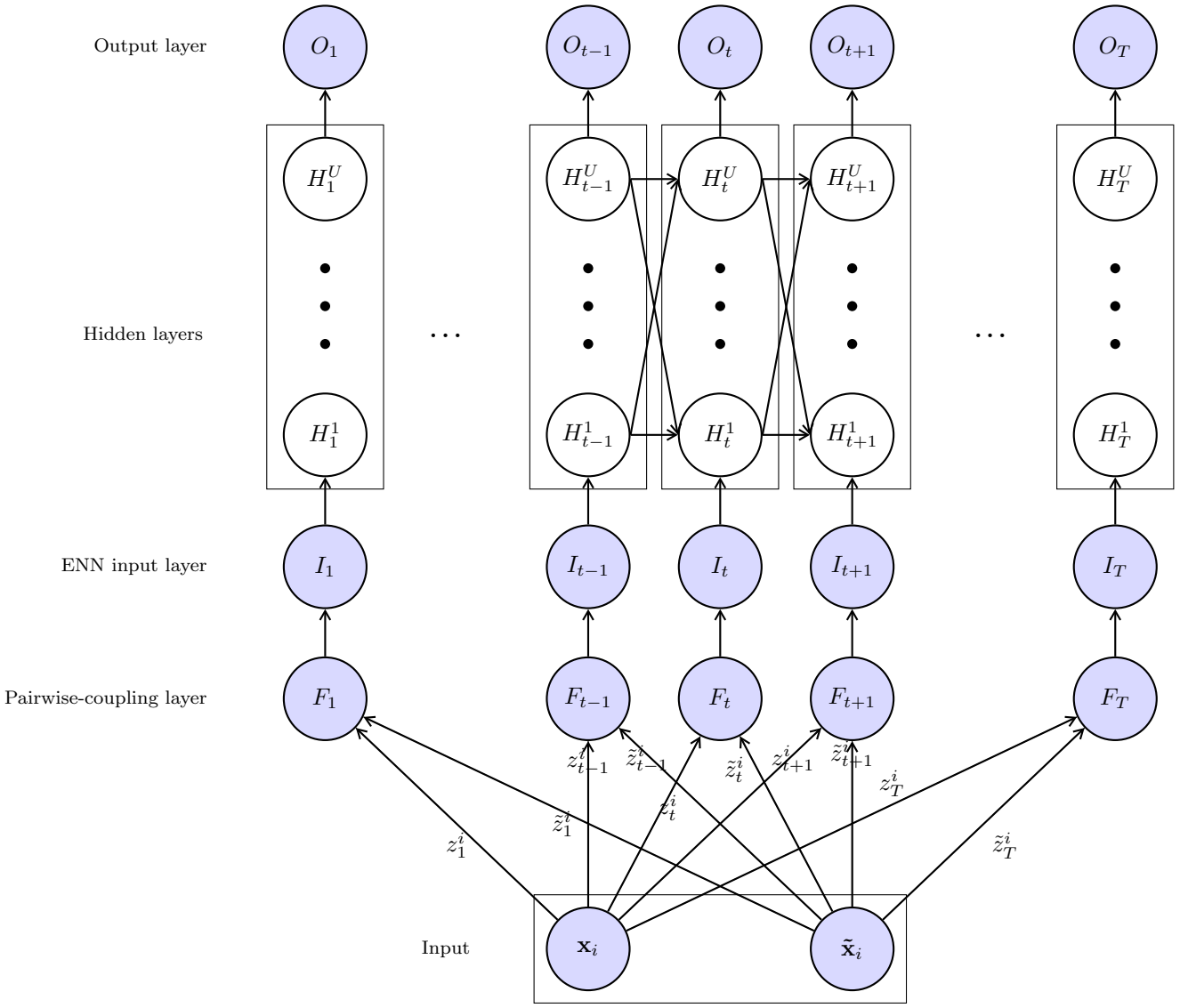
$$
\begin{aligned}
&\boldsymbol{\Sigma} - (\boldsymbol{\Sigma} - diag(\mathbf{s}))\boldsymbol{\Sigma}^{-1}(\boldsymbol{\Sigma} - diag(\mathbf{s})) \\
&= \boldsymbol{\Sigma} - (\boldsymbol{\Sigma}\boldsymbol{\Sigma}^{-1} - diag(\mathbf{s})\boldsymbol{\Sigma}^{-1})(\boldsymbol{\Sigma} - diag(\mathbf{s})) \\
&= \boldsymbol{\Sigma} - (\boldsymbol{\Sigma}\boldsymbol{\Sigma}^{-1}\boldsymbol{\Sigma} - \boldsymbol{\Sigma}\boldsymbol{\Sigma}^{-1}diag(\mathbf{s}) - diag(\mathbf{s})\boldsymbol{\Sigma}^{-1}\boldsymbol{\Sigma} + diag(\mathbf{s})\boldsymbol{\Sigma}^{-1}diag(\mathbf{s})) \\
&= \boldsymbol{\Sigma} - (\mathbf{I}\boldsymbol{\Sigma} - \mathbf{I}diag(\mathbf{s}) - diag(\mathbf{s})\mathbf{I} + diag(\mathbf{s})\boldsymbol{\Sigma}^{-1}diag(\mathbf{s})) \\
&= \boldsymbol{\Sigma} - \boldsymbol{\Sigma} + 2diag(\mathbf{s}) - diag(\mathbf{s})\boldsymbol{\Sigma}^{-1}diag(\mathbf{s}) \\
&= 2diag(\mathbf{s}) - diag(\mathbf{s})\boldsymbol{\Sigma}^{-1}diag(\mathbf{s}).
\end{aligned}
$$

s is the solution to the maximization problem: $2\boldsymbol{\Sigma} - diag(\mathbf{s}) > 0$, $0 \leq s \leq 1$. This is to ensure that matrix $\mathbf{A}$ is positive definite as a positive definite matrix also has a positive determinant. For matrix $\mathbf{A}$ the determinant must be bigger than zero. Hence, the following must hold:

$$
\begin{aligned}
&\boldsymbol{\Sigma}\boldsymbol{\Sigma} - (\boldsymbol{\Sigma} - diag(\mathbf{s}))(\boldsymbol{\Sigma} - diag(\mathbf{s})) > 0 \\
\Longleftrightarrow &\boldsymbol{\Sigma}\boldsymbol{\Sigma} - \boldsymbol{\Sigma}\boldsymbol{\Sigma} + 2\boldsymbol{\Sigma}diag(\mathbf{s}) - diag(\mathbf{s})diag(\mathbf{s}) > 0 \\
\Longleftrightarrow &2\boldsymbol{\Sigma}diag(\mathbf{s}) - diag(\mathbf{s})diag(\mathbf{s}) > 0 \\
\Longleftrightarrow &(2\boldsymbol{\Sigma} - diag(\mathbf{s}))diag(\mathbf{s}) > 0 \\
\Longleftrightarrow &2\boldsymbol{\Sigma} - diag(\mathbf{s}) > 0.
\end{aligned}
$$

The knockoffs are created using the knockoff package, where the maximization of s is done using semidefinite programming with the package Rdsdp (R Core Team, 2019). All k generated knockoff variables will be taken and the p relevant knockoffs will be selected at the end of each epoch during the training process after running the knockoffs and original variables in an ENN. The knockoff variables and original variables will be sorted in a decreasing manner using the absolute knockoff statistics, before combining these variables as input in an ENN. The approach in this paper differs from the approach done in Lu et al. (2018), where Lu et al. (2018) already selected the p relevant knockoffs after making them and then used these knockoffs inside an FNN.

Graphically the fully connected ENN with a DeepPINK framework and model-X knockoffs would look as follows for variable and knockoff variable i:



The $N \times 1$ vectors $\mathbf{x}_i$ and $\tilde{\mathbf{x}}_i$ $(i = 1, ..., k)$ are respectively vectors for the original features (variables) and the corresponding knockoff variables with scalar filter weights of respectively z_i and $\tilde{z}_i$. Each pairwise couple is an $N \times 1$ vector F_t that forms a node in the pairwise-coupling layer. Here F_t for update step t is only one variable in the pairwise-coupling layer and more nodes are present when using more than one variable in this model for each update step t of the training process. The algorithm for an ENN with integrated knockoff filters is formulated in Algorithm 1. The proposed model will be referred to as 'EDP' in this paper. The configuration and some terminology of the training process in this study are given in Appendix B.

The derivation of the feedforward part in Algorithm 1 from the ENN input layer and further is based on the mathematical derivation in the study of Pham and Liu (1996) and Chong et al. (2017), where the output of each hidden layer is influenced by all past and current hidden nodes (Kuan, 2006). The bias term always has a value of 1 (Rumelhart et al., 1986). For this study I did not implement a bias term in EDP in line with the FNN with DeepPINK in Lu et al. (2018).

The initial values in the $k \times 1$ filter weight vector $\mathbf{z}$ and the $k \times 1$ knockoff filter weight vector $\tilde{\mathbf{z}}$, as well as the initial values for all weights in every layer in the model, are randomly set in

Algorithm 1 Using the original variables and model-X knockoffs inside an Elman neural network

Precondition: The variables in the $N \times k$ matrix $\mathbf{X}$ are multivariate Guassian

Make an $N \times k$ knockoffs matrix $\tilde{\mathbf{X}}$ from the original input $\mathbf{X}$
for $e \leftarrow 1$ to E **do** ▷ E: number of epochs
  for $f \leftarrow 1$ to F **do** ▷ F: number of folds

    Initialize weights

    Start training process ▷ see Appendix B
    while $t <$ *updates* **do** ▷ *updates*: number of updates in the training process
      Start forward pass
      for $i \leftarrow 1$ to k **do** ▷ k: number of variables
        $F_t^i \leftarrow z_t^i \mathbf{x}_i + \tilde{z}_t^i \tilde{\mathbf{x}}_i$ ▷ Pairwise-coupling layer
        $G_t^i \leftarrow f(F_t^i)$ ▷ $f(\cdot)$: ReLU function
      end for
      for $i \leftarrow 1$ to k **do**
        $R_t^i \leftarrow \mathbf{w}_{0,t} \odot G_t^i$ ▷ ENN input layer
        $I_t^i \leftarrow f(R_t^i)$
      end for
      for $i \leftarrow 1$ to k **do**
        for $j \leftarrow 1$ to *numvar* **do**
          $\nu_t^{1,i} \leftarrow \sum_{j=1}^{p} w_{1,t-1}^{j,i} I_t^j + b_1 + H_{t-1}^{1,j}$ ▷ 1^{st} hidden layer, $b_\square$: bias term
          $H_t^{1,i} \leftarrow f(\nu_t^{1,i})$
        end for
      end for
      if $U = 2$ **then** ▷ U: number of hidden layers
        for $i \leftarrow 1$ to k **do**
          for $j \leftarrow 1$ to *numvar* **do**
            $\nu_t^{u,i} \leftarrow \sum_{j=1}^{p} w_{u,t-1}^{j,i} H_t^{u-1,j} + b_u + H_{t-1}^{u,i}$ ▷ 2^{nd} hidden layer, $u = 2$
            $H_t^{u,i} \leftarrow f(\nu_t^{u,i})$
          end for
        end for
      end if
      for $i \leftarrow 1$ to k **do**
        $O_t \leftarrow \sum_{i=1}^{k} w_{U+1,t-1}^{i} H_t^{U,i}$ ▷ Output layer
      end for
      End forward pass

      Backpropagation ▷ see Appendix C
    end while
    End training process

    Calculate feature importance measures and knockoff statistics after each batch ▷ See Section 2.3.2

    Selecting the relevant knockoffs

    Use the same algorithm as the forward pass for the test set
  end for
end for

the range of (-0.5, 0.5) with half of the weights within the range (-0.5, 0) and half within the range (0, 0.5). The weights are initialized this way as otherwise the outcome of the weights may get too large or too low after convergence during the training process. The first run is always a forward pass, such that there are no updates yet in the RNN.

The nodes in the input layer after the pairwise-coupling layer are not determined by the sum of nodes of the previous layer, but each filtered input is connected separately to one node in the ENN input layer. As the $k \times 1$ vector $\mathbf{w}_{0,t}$ contains the weights after the pairwise-coupling layer, the number of weights is the same as the number of variables, given that the DeepPINK filter combines k true variables and k knockoffs into k filtered input nodes. In line with Pham and Liu (1996) I decide not to use an activation function in the output layer.

2.3.1 Regularization of the weights

The standard backpropagation algorithm (Rumelhart et al., 1986) will be implemented for updating the weights after each training update using gradient descent (see Appendix C) as this is considered to be a robust approach for supervised mapping, where the term mapping refers to finding the parameters, given the input and desired output. The error function will be used to derive the gradient (derivative) through backpropagation (Chong et al., 2017; Cao, Wang, Ming, & Gao, 2017).

The error from the backpropagation will be calculated by taking the quadratic cost function (or loss function) in line with Pham and Liu (1996) and Nabian and Meidani (2019). The quadratic error will be calculated as:

$$E_t = \frac{1}{2}(y_{train} - O_t)^2, \tag{2.8}$$

where y_{train} is the desired output in the training set and O_t is the calculated output at update step t in the training process.

During the updates of the weights in the training process the context units may also become unstable or saturated, which means that the weights need to be controlled during the training process (Koskela, 2003). Given that the least absolute shrinkage and selection operator (LASSO) L1-regularization is commonly used for controlling weights when using NNs for feature selection purposes (Nagpal, 2017), this paper will use L1-regularization as weight decay method in combination with the loss function in line with Lu et al. (2018). The regularization will be conducted by taking the cost function:

$$C = E_t^{L_1} = E_t + \lambda ||\mathbf{W}_t||, \tag{2.9}$$

where E_t is the loss function at update step t during training, $\lambda \in [0, \infty)$ is a hyperparameter used as penalty term to prevent the weights for being too large and where $||\mathbf{W}||$ is the length of $\mathbf{W} = \{\mathbf{w}_0, \mathbf{w}_1, \ldots, \mathbf{w}_U, \mathbf{w}_{U+1}\}$. Here $||\mathbf{W}||$ is considered to be the norm of $||\mathbf{W}||$ (Nabian & Meidani, 2019). This norm can be defined by the expression $\sqrt{\langle \mathbf{W}, \mathbf{W} \rangle} = \sqrt{\mathbf{W} \odot \mathbf{W}} = |\mathbf{W}| =$

$\sum_{i,j} |w_{i,j}|$, where $\langle \cdot, \cdot \rangle$ is the inner product, $\odot$ is the *dot*-product (or element-wise multiplication of the matrices) and $|\cdot|$ is the absolute, not to confuse with the sign for the determinant of a matrix. Nabian and Meidani (2019) set the penalty term to 0.5, which will also be the choice in this study. The higher the penalty term will be set, the lower the calculated weights and the lower the variance (Nusrat & Jang, 2018). In case bias terms were used in the model, note that only the weights would be subject to regularization and the bias terms in the model would remain unregularized (Nabian & Meidani, 2019).

As the L1-regularization scales the additional term with a constant term this allows for sparsity in the weights which means that more of the weights are shrunk to zero (Nabian & Meidani, 2019; Foygel Barber & J. Candès, 2015). In other words more nodes are getting a weight of zero in the model after each update as means of removing irrelevant characteristics from the model. This regularization method differs from the dropout regularization technique (Nabian & Meidani, 2019), which removes nodes randomly which will not have an effect during the following update.

2.3.2 Feature importance

The network basically learns through updating the weights during training (Gradojevic & Yang, 2000; Lobbrecht et al., 2005). The more epochs, the more accurate the outcomes should become and the smaller the error. From the second update till the last update the hidden layer is updated recurrently, where each recurrent hidden layer node has a weight of 1 (Elman, 1990). The 1×1 filter weights z_i and $\tilde{z}_i$ will compete against each other during the training process and the $k \times 1$ feature importance measures $\mathbf{d}$ and $\tilde{\mathbf{d}}$ will be determined as follows (Lu et al., 2018), where the last updated weights (update T) are used:

$$\begin{aligned} d_i &= v_i z_i, \\ \tilde{d}_i &= v_i \tilde{z}_i, \text{ for } i = 1, ..., k \text{ features}, \end{aligned} \tag{2.10}$$

where the scalar v_i is the i-th element of $\nu = \mathbf{w}_{0,T} \odot \left((\prod_{u=1}^{U} \mathbf{w}_{u,T}) \mathbf{w}_{U+1,T} \right)$ with U the number of hidden layers and $\mathbf{w}_{U+1}$ the vector with the output weights. In case no DeepPINK framework is applied, then only v_i is taken, where only the knockoffs are used in an ENN model framework. This choice can be made as the calculated weights under an L1-regularization are minimized and as the importance measures are then determined by the relative importance between the true variables and the knockoff variables, which makes it possible to use this calculated weight matrix as a relative importance measure also in the case where no pairwise-coupling layer is assumed.

With the updated importance measures, the knockoff statistics will be calculated at the end of every epoch (as seen in Algorithm 1) in order to see if the relevance of the features (variables) change. Calculating the knockoff statistics after every epoch will allow the filter weights to compete against each other during training. All variables are used at every update during the training process. The knockoff statistics are calculated after all updates within an epoch have been completed in order to check per epoch which variables are assumed to be relevant.

As said earlier, the knockoff statistics will have an anti-symmetry property, which means that, when the i-th feature importance measure is swapped with its knockoff counterpart, this will only change the sign of the outcome of the i-th knockoff statistic l_i. When the value of l_i is large and positive, this means that the true feature is considered to be relevant for explaining the outcomes of the model, while it is not considered to be relevant otherwise. The knockoff statistics will be calculated by $l_i(\mathbf{X}, \tilde{\mathbf{X}}, \mathbf{y}) = |d_i| - |\tilde{d}_i|$ (Foygel Barber & J. Candès, 2016) as I want to cancel out the effect of a negative value for the importance measures and want to see the effect of the measures, regardless of the sign of these importance measures. I will not take the absolute value of the calculated knockoff statistic when no DeepPINK filter is implemented, otherwise all knockoff statistics would be positive. Both the knockoff statistics and the feature importance measures will vary when varying the algorithm of the model (Lu et al., 2018). The knockoff statistics will be updated during the training process, where they will change with the change of the updated weights at the end of each epoch.

2.4 Comparison with other models

In this study the prediction performance of the Elman Network combined with the model-X knockoffs framework and the DeepPINK framework will be compared against:

- The Elman Network with model-X knockoffs without a DeepPINK model framework.
- FNN with model-X knockoffs and DeepPINK.

2.4.1 Performance metrics

One wants to find a set of weights which minimizes the distance between fitted values and the values in the training set. If the model performs well, the updated weight should lie closely to the old weight after each update. Also, the sign should not change (Sun, 1999). For measuring the prediction performance, one can look at three measures: (1) Calibration measures in order to test for a bias in the predictions; (2) Measures of overall performance to examine the difference between the predictions and the true values; (3) Measures of model discrimination which test for the ability of the model to separate different outcomes classes (Steyerberg et al., 2010).

In the test set the true $\mathbf{y}$ variables will be taken as benchmark of performance. As a measure for overall performance, the mean squared prediction error (MSPE) (equation (2.13)) will be used to test for prediction accuracy in the test set as this measure is usually applicable in cases where prediction performance is tested in datasets for which no robust estimation is necessary. Here lies the assumption that the dataset is not subject to excessive outliers. In case there are outliers, the MSPE can get inflated. The mean absolute prediction error (MAPE) may then be a better choice as a prediction performance measure. However, the MAPE still has the drawback that the same weights are given to all observations in the calculation of the measure. Hence, the MSPE will be used and will be calculated as:

$$MSPE_{f,e} = \frac{1}{N-S} \sum_{i=1}^{N-S} (y_{i,test} - \hat{y}_{i,test})^2, \tag{2.11}$$

where $\hat{\square}$ is the estimated outcome, N is the total samples (instances) in the dataset, S is the number of samples in the training set and $N - S$ is the number of samples in the test set. This is the MSPE for fold f and for epoch e in the training process.

The MSPE is calculated over each fold by taking the mean over all E epochs per fold of the $MSPE_{f,e}$ measure (equation (2.12)). Subsequently, the mean over all F folds will be taken in order to get one MSPE outcome. In line with Lu et al. (2018) the feature importance will be compared between the different models by examining the knockoff statistics and the number of selected features.

$$MSPE_f = \frac{1}{E} \sum_{e=1}^{E} MSPE_{f,e}, \tag{2.12}$$

$$MSPE = \frac{1}{F} \sum_{f=1}^{F} MSPE_f, \tag{2.13}$$

Note that one cannot calculate measures which use the Likelihood, e.g. the AIC or the BIC, as NNs do not make use of a Likelihood function to estimate the parameters. It will be possible to look at the mean squared error (or any other form of it like the root mean squared error) during a simulation study, where benchmark parameters can be introduced, which allows to compare these benchmark parameters to the estimated weights.

3 Results

3.1 Simulation design

In order to test for accuracy a simulation study will be conducted. Given the mathematical workload that is needed to run the EDP and the other NNs, ENN and FNN with a DeepPINK filter, the simulation will be repeated 2 times. The default value for creating the folds for the training set is set to 2 folds, the number of batches that are created, is 2 and the number of epochs is 2. Data with 60 observations and 30 features will be generated using a data generating process (DGP), where this DGP reflects the true population. The objective of the simulation study is to verify to what extent the EDP can find the true parameter values in a simulated environment. Hence, the population serves as a benchmark for data that will be contaminated, such that one can see how the ENN can be improved with the use of the EDP model and how the estimations will alter. Note that the higher the values are set the more reliable the outcomes would be. In this study I have chosen these default values due to the high mathematical workload it would take to run all simulations. As DGP the following model will be chosen:

$$\mathbf{y} = \mathbf{X}\beta + \epsilon, \tag{3.1}$$

where $\mathbf{y} \sim \mathcal{N}(\mathbf{X}\beta, \mathbf{I}) \in \mathbb{R}^N$ is the output vector, $\beta \in \mathbb{R}^k$ is the parameter vector, $\epsilon \sim \mathcal{N}(0, 1)$, the features $\mathbf{x}_i$ for $i = 1, ..., k$ are randomly generated such that the matrix $\mathbf{X} \in \mathbb{R}^{N \times k}$, containing all k features, are Gaussian and $\mathbf{X} \sim \mathcal{N}(0, \mathbf{I})$. This DGP will be subjected to contamination in the data, which will be introduced by using different feature correlations and different levels of non-zero signals in the parameter vector (Foygel Barber & J. Candès, 2015).

The knockoff features will be generated in a way that $\tilde{\mathbf{X}} \sim \mathcal{N}(0, \tilde{\mathbf{\Sigma}}_\gamma)$, where $\tilde{\sigma}_{\gamma,i,j} = \gamma^{|i-j|}$ for correlation levels $\gamma = 0, 0.1, ..., 0.9$. In case $\gamma = 0$, $\tilde{\mathbf{\Sigma}}_0 = \mathbf{I}$, the identity matrix (Foygel Barber & J. Candès, 2015). The true DGP values for the dependent variable $\mathbf{y}$ are derived by using the true features $\mathbf{X}$. The dependent variable $\mathbf{y}$ will then be used together with the generated knockoff feature vectors $\tilde{\mathbf{x}}_i$ in the models to estimate the parameters. The estimated coefficients will then be compared with the true DGP coefficients.

To introduce sparsity, true DGP coefficients β are generated by randomly allocating non-zero signals from a sequence of 10 to 30 signals with steps of 10 within the simulation. Places in β which are not getting a non-zero signal are kept 0. The non-zero signals for the true DGP coefficients are randomly set within the range (-1.5, 1.5) in line with Lu et al. (2018). For the knockoffs the FDR level is fixed at 0.20 (= 20%). Testing for several correlations between the features and for several sparsity levels is crucial to see if alternating the correlations between

the features matter in selecting the features and to investigate if having zero signals leads to substantially different parameter estimations.

As measure for accuracy of the estimated weights the mean squared error (MSE) will be used, which can be formulated as:

$$MSE_f = \frac{1}{R}\sum_{r=1}^{R}(\beta_{DGP,type,r} - \hat{\beta}_{test,type,r})^2, \tag{3.2}$$

with R the number of simulation runs, $type = \{10, 20, 30\}$ for the number of non-zero signals and $f = 1, ...F$ folds.

For the estimated weights the mean of the knockoff statistics over all epochs are taken for each test set (equation (3.3) and equation (3.4)). The knockoff statistics are, hence, used as measures for the relative importance of each feature.

$$\hat{\beta}_{test,type,e,r} = l_i(\mathbf{X}, \tilde{\mathbf{X}}, \mathbf{y})_{test,type,e,r}, \text{ for } e = 1, ..., E \text{ epochs and } r = 1, ..., R \text{ simulations.} \tag{3.3}$$

$$\hat{\beta}_{test,type,r} = \frac{1}{E}\sum_{e=1}^{E}\hat{\beta}_{test,type,e,r}. \tag{3.4}$$

This choice is made because for the feature importance measures the importance between the links inside an ENN (and the filter weights in case the DeepPINK filter is implemented) is used, which can then be seen as a measure for the relative importance of a feature for explaining the output, which can then also be interpreted as the total weight or the β coefficient of that particular feature for explaining the output variable.

These coefficients are subtracted from the true DGP coefficients, the square is taken from this outcome, after which the mean over all simulations are taken to get the MSE per test set (equation (2.13)). The mean is taken over the MSE of all test sets, $\frac{1}{F}\sum_{f=1}^{F} MSE_f$, in order to get one MSE outcome per non-zero signal level and correlation level.

Call the number of selected knockoff variables $p_{f,type,e,r}$ for the number of selected variables after each training process for a number of non-zero signals used at epoch e and simulation r. These variables are selected within each examined model. The mean over the number of selected variables after all epochs per training set is taken, $p_{f,type,r} = \frac{1}{E}\sum_{e=1}^{E} p_{f,type,e,r}$, after which the mean is taken over all simulations for each training set, $p_{f,type} = \frac{1}{r}\sum_{r=1}^{R} p_{f,type,r}$, to get the number of selected variables for each training set. The ceiling value is then taken for the mean over all training sets, $p_{type} = \lceil \frac{1}{f}\sum_{f=1}^{F} p_{f,type} \rceil$, in order to prevent the method for selecting less relevant variables than possible as selecting less relevant variables can lead to parsimony, which could bias the results into getting preferred outcomes.

$$MSPE_{f,e,r} = \frac{1}{N-S}\sum_{i=1}^{N-S}(y_{i,test,r} - \hat{y}_{i,test,r})^2, \tag{3.5}$$

$$MSPE_{f,r} = \frac{1}{E} \sum_{e=1}^{E} MSPE_{f,e,r}, \tag{3.6}$$

For the MSPE measures, the measure is first calculated per fold f and per epoch e at simulation run r (equation (3.5)), after which the mean for this calculated measure is taken over all the epochs per fold to get to the MSPE for each fold at simulation r (equation (3.6)). Subsequently, the mean is taken over the R simulations in order to get the MSPE for each fold over all simulations, denoted by $MSPE_{f,sim} = \frac{1}{R} \sum_{r=1}^{R} MSPE_{f,r}$. Lastly, the mean is taken over each fold to get one MSPE measure $MSPE_{sim} = \frac{1}{F} \sum_{f=1}^{F} MSPE_{f,sim}$.

3.1.1 Simulation results

Table 3.1: Simulation results for one hidden layer and the modified FDR

This table gives the statistical results for a simulation study with a 2-fold cross-validation. The number of features that are taken is 30 with 3 sparsity levels. Each training set and test set are generated using 2-fold cross-validation. The simulation is done 2 times, where each training set is being split into 2 batches and where the number of epochs is 2. The outcomes are given for an FNN and an ENN with 1 hidden layer, where the modified FDR is being controlled.									
ENN, 1 hidden layer, modified FDR is controlled									
	MSPE			MSE			Number of selected features		
feature correlation	non-zero signals			non-zero signals			non-zero signals		
	10	20	30	10	20	30	10	20	30
0	11.403	17.270	26.727	0.885	0.900	0.909	30	30	30
0.1	9.136	17.147	26.937	0.887	0.896	0.895	30	30	30
0.2	9.468	17.242	26.905	0.889	0.893	0.906	30	30	30
0.3	9.483	17.229	26.991	0.889	0.896	0.903	30	30	30
0.4	9.483	17.393	26.880	0.894	0.903	0.902	30	30	30
0.5	9.610	17.709	27.177	0.894	0.900	0.903	30	30	30
0.6	9.664	17.686	27.200	0.896	0.904	0.904	30	30	30
0.7	9.480	17.239	26.998	0.902	0.910	0.905	30	30	30
0.8	9.849	17.787	27.036	0.910	0.916	0.909	30	30	30
0.9	9.851	17.916	27.317	0.904	0.918	0.909	30	30	30
EDP, 1 hidden layer, modified FDR									
0	11.198	20.125	24.836	0.870	0.675	0.675	28	28	28
0.1	9.335	20.134	24.775	0.674	0.675	0.675	28	28	28
0.2	9.261	20.124	24.876	0.674	0.675	0.675	28	28	28
0.3	9.263	20.175	24.853	0.674	0.675	0.675	28	28	28
0.4	9.268	20.145	24.829	0.674	0.675	0.675	28	28	28
0.5	9.281	20.157	24.866	0.674	0.675	0.675	28	28	28
0.6	9.251	20.134	24.825	0.674	0.675	0.675	28	28	28
0.7	9.255	20.125	24.803	0.674	0.675	0.675	28	28	28
0.8	9.266	20.153	24.856	0.674	0.675	0.674	28	28	28
0.9	9.330	20.174	24.834	0.674	0.675	0.675	28	28	28
FNN with DeepPINK filter, 1 hidden layer, modified FDR									
0	11.202	20.129	24.849	0.870	0.674	0.675	28	28	28
0.1	9.336	20.135	24.785	0.674	0.674	0.674	28	28	28
0.2	9.262	20.122	24.883	0.674	0.674	0.674	28	28	28
0.3	9.263	20.170	24.857	0.674	0.674	0.675	28	28	28
0.4	9.269	20.149	24.841	0.674	0.674	0.674	28	28	28
0.5	9.282	20.158	24.874	0.674	0.674	0.674	28	28	28
0.6	9.252	20.134	24.833	0.674	0.674	0.674	28	28	28
0.7	9.254	20.122	24.809	0.674	0.674	0.674	28	28	28
0.8	9.266	20.150	24.865	0.674	0.674	0.674	28	28	28
0.9	9.330	20.171	24.845	0.674	0.674	0.674	28	28	28

Table 3.2: Simulation results for two hidden layers and the modified FDR

This table gives the statistical results for a simulation study with a 2-fold cross-validation. The number of features that are taken is 30 with 3 sparsity levels. Each training set and test set are generated using 2-fold cross-validation. The simulation is done 2 times, where each training set is being split into 2 batches and where the number of epochs is 2. The outcomes are given for an FNN and an ENN with 2 hidden layers, where the modified FDR is being controlled.

ENN, 2 hidden layers, modified FDR is controlled									
	MSPE			MSE			Number of selected features		
feature correlation	non-zero signals			non-zero signals			non-zero signals		
	10	20	30	10	20	30	10	20	30
0	14.454	15.906	26.731	0.922	0.773	0.776	29	30	29
0.1	8.699	16.803	28.155	0.779	0.780	0.775	29	29	29
0.2	8.169	16.665	28.594	0.777	0.772	0.773	29	29	29
0.3	7.292	15.379	26.013	0.772	0.781	0.778	29	29	29
0.4	6.659	16.051	26.562	0.775	0.769	0.771	29	29	29
0.5	6.642	15.366	25.924	0.772	0.775	0.776	29	29	29
0.6	6.098	15.005	25.708	0.777	0.777	0.776	30	29	29
0.7	7.816	15.891	28.702	0.780	0.779	0.787	29	29	29
0.8	7.513	16.245	28.954	0.779	0.774	0.785	29	29	29
0.9	6.472	14.746	26.320	0.781	0.775	0.777	30	29	29
EDP, 2 hidden layers, modified FDR									
0	11.278	16.966	24.083	0.878	0.910	0.911	30	30	30
0.1	7.981	17.057	24.079	0.910	0.910	0.911	30	30	30
0.2	7.964	17.000	24.044	0.910	0.910	0.911	30	30	30
0.3	7.967	17.007	24.020	0.910	0.910	0.911	30	30	30
0.4	7.945	17.006	23.913	0.910	0.910	0.911	30	30	30
0.5	7.986	17.035	23.997	0.910	0.910	0.911	30	30	30
0.6	7.993	17.000	23.989	0.910	0.910	0.911	30	30	30
0.7	8.066	17.090	24.128	0.910	0.910	0.911	30	30	30
0.8	8.032	17.058	24.150	0.910	0.910	0.911	30	30	30
0.9	8.044	17.014	24.011	0.910	0.910	0.911	30	30	30
FNN with DeepPINK filter, 2 hidden layers, modified FDR									
0	11.260	16.965	24.177	0.8780	0.909	0.909	30	30	30
0.1	7.975	17.063	24.152	0.909	0.909	0.909	30	30	30
0.2	7.952	17.004	24.117	0.909	0.909	0.909	30	30	30
0.3	7.971	17.019	24.131	0.909	0.909	0.909	30	30	30
0.4	7.943	17.017	24.057	0.909	0.909	0.909	30	30	30
0.5	7.987	17.032	24.087	0.909	0.909	0.909	30	30	30
0.6	8.003	16.997	24.099	0.909	0.909	0.909	30	30	30
0.7	8.053	17.096	24.235	0.909	0.909	0.909	30	30	30
0.8	8.031	17.069	24.228	0.909	0.909	0.909	30	30	30
0.9	8.054	17.026	24.140	0.909	0.909	0.909	30	30	30

Table 3.3: Simulation results for one hidden layer and the exact FDR

This table gives the statistical results for a simulation study with a 2-fold cross-validation. The number of features that are taken is 30 with 3 sparsity levels. Each training set and test set are generated using 2-fold cross-validation. The simulation is done 2 times, where each training set is being split into 2 batches and where the number of epochs is 2. The outcomes are given for an FNN and an ENN with 1 hidden layer, where the exact FDR is being controlled.

ENN, 1 hidden layer, exact FDR is controlled									
	MSPE			MSE			Number of selected features		
feature correlation	non-zero signals			non-zero signals			non-zero signals		
	10	20	30	10	20	30	10	20	30
0	11.403	17.270	26.727	0.885	0.900	0.909	30	30	30
0.1	9.136	17.147	26.937	0.887	0.896	0.895	30	30	30
0.2	9.468	17.242	26.905	0.889	0.893	0.906	30	30	30
0.3	9.483	17.229	26.991	0.889	0.896	0.903	30	30	30
0.4	9.483	17.393	26.880	0.894	0.903	0.902	30	30	30
0.5	9.610	17.709	27.177	0.894	0.900	0.903	30	30	30
0.6	9.664	17.686	27.200	0.896	0.904	0.904	30	30	30
0.7	9.480	17.239	26.998	0.902	0.910	0.905	30	30	30
0.8	9.849	17.787	27.036	0.910	0.916	0.909	30	30	30
0.9	9.851	17.916	27.317	0.904	0.918	0.909	30	30	30
EDP, 1 hidden layer, exact FDR									
0	11.198	20.125	24.836	0.870	0.675	0.675	30	30	30
0.1	9.335	20.134	24.775	0.674	0.675	0.675	30	30	30
0.2	9.261	20.124	24.876	0.674	0.675	0.675	30	30	30
0.3	9.263	20.175	24.853	0.674	0.675	0.675	30	30	30
0.4	9.268	20.145	24.829	0.674	0.675	0.675	30	30	30
0.5	9.281	20.157	24.866	0.674	0.675	0.675	30	30	30
0.6	9.251	20.134	24.825	0.674	0.675	0.675	30	30	30
0.7	9.255	20.125	24.803	0.674	0.675	0.675	30	30	30
0.8	9.266	20.153	24.856	0.674	0.675	0.674	30	30	30
0.9	9.330	20.174	24.834	0.674	0.675	0.675	30	30	30
FNN with DeepPINK filter, 1 hidden layer, exact FDR									
0	11.202	20.129	24.849	0.870	0.674	0.675	30	30	30
0.1	9.336	20.135	24.785	0.674	0.674	0.674	30	30	30
0.2	9.262	20.122	24.883	0.674	0.674	0.674	30	30	30
0.3	9.263	20.170	24.857	0.674	0.674	0.675	30	30	30
0.4	9.269	20.149	24.841	0.674	0.674	0.674	30	30	30
0.5	9.282	20.158	24.874	0.674	0.674	0.674	30	30	30
0.6	9.252	20.134	24.833	0.674	0.674	0.674	30	30	30
0.7	9.254	20.122	24.809	0.674	0.674	0.674	30	30	30
0.8	9.266	20.150	24.865	0.674	0.674	0.674	30	30	30
0.9	9.330	20.171	24.845	0.674	0.674	0.674	30	30	30

Table 3.4: Simulation results for two hidden layers and the exact FDR

This table gives the statistical results for a simulation study with a 2-fold cross-validation. The number of features that are taken is 30 with 3 sparsity levels. Each training set and test set are generated using 2-fold cross-validation. The simulation is done 2 times, where each training set is being split into 2 batches and where the number of epochs is 2. The outcomes are given for an FNN and an ENN with 2 hidden layers, where the exact FDR is being controlled.

ENN, 2 hidden layers, exact FDR is controlled									
	MSPE			MSE			Number of selected features		
feature correlation	non-zero signals			non-zero signals			non-zero signals		
	10	20	30	10	20	30	10	20	30
0	14.454	15.906	26.731	0.922	0.773	0.776	30	30	30
0.1	8.699	16.803	28.155	0.779	0.780	0.775	30	30	30
0.2	8.169	16.665	28.594	0.777	0.772	0.773	30	30	30
0.3	7.292	15.379	26.013	0.772	0.781	0.778	30	30	30
0.4	6.659	16.051	26.562	0.775	0.769	0.771	30	30	30
0.5	6.642	15.366	25.924	0.772	0.775	0.776	30	30	30
0.6	6.098	15.005	25.708	0.777	0.777	0.776	30	30	30
0.7	7.816	15.891	28.702	0.780	0.779	0.787	30	30	30
0.8	7.513	16.245	28.954	0.779	0.774	0.785	30	30	30
0.9	6.472	14.746	26.320	0.781	0.775	0.777	30	30	30
EDP, 2 hidden layers, exact FDR									
0	11.278	16.966	24.083	0.878	0.910	0.911	30	30	30
0.1	7.981	17.057	24.079	0.910	0.910	0.911	30	30	30
0.2	7.964	17.000	24.044	0.910	0.910	0.911	30	30	30
0.3	7.967	17.007	24.020	0.910	0.910	0.911	30	30	30
0.4	7.945	17.006	23.913	0.910	0.910	0.911	30	30	30
0.5	7.986	17.035	23.997	0.910	0.910	0.911	30	30	30
0.6	7.993	17.000	23.989	0.910	0.910	0.911	30	30	30
0.7	8.066	17.090	24.128	0.910	0.910	0.911	30	30	30
0.8	8.032	17.058	24.150	0.910	0.910	0.911	30	30	30
0.9	8.044	17.014	24.011	0.910	0.910	0.911	30	30	30
FNN with DeepPINK filter, 2 hidden layers, exact FDR									
0	11.260	16.965	24.177	0.878	0.909	0.909	30	30	30
0.1	7.975	17.063	24.152	0.909	0.909	0.909	30	30	30
0.2	7.952	17.004	24.117	0.909	0.909	0.909	30	30	30
0.3	7.971	17.019	24.131	0.909	0.909	0.909	30	30	30
0.4	7.943	17.017	24.057	0.909	0.909	0.909	30	30	30
0.5	7.987	17.032	24.087	0.909	0.909	0.909	30	30	30
0.6	8.003	16.997	24.099	0.909	0.909	0.909	30	30	30
0.7	8.053	17.096	24.235	0.909	0.909	0.909	30	30	30
0.8	8.031	17.069	24.228	0.909	0.909	0.909	30	30	30
0.9	8.054	17.026	24.140	0.909	0.909	0.909	30	30	30

Overall, these results show that the MSPE and the MSE increase when the non-zero signals increase. The MSE does decrease when using an ENN with 2 hidden layers (Table 3.2 and Table 3.4), though, when the non-zero signals increase. These outcomes imply that the range that is chosen for the true DGP coefficients is higher than the estimated parameters (which fluctuate closer around zero) in case of an increase of the errors when the non-zero signals increase.

There is no clear pattern noticeable in the MSE and MSPE when the feature correlations rise, implying that varying the feature correlations would not clearly lead to better or worse feature selection. One would expect that a high feature correlation would lead to less features being selected using the proposed model and, thus to a better subset selection. The models with DeepPINK seem to give less precise outcomes concerning the higher MSPE and MSE than the ENN without DeepPINK when 2 hidden layers are used (Table 3.2 and Table 3.4).

Regarding feature selection with DeepPINK and one hidden layer, it seems that less relevant features are selected when the modified FDR measure is being used as threshold (Table 3.1) for selecting the relevant features compared to the exact FDR (Table 3.3) that was introduced by Benjamini and Hochberg (1995). The use of 1 hidden layer and using DeepPINK and the modified FDR threshold give lower MSE compared to when the models with 2 hidden layers are being used (Table 3.2), where the EDP and FNN with DeepPINK select the same number of variables and perform both approximately the same in terms of MSPE and MSE (Table 3.1) and where the ENN has lower MSE and MSPE values for 2 hidden layers than an ENN with one hidden layer (Table 3.1 vs Table 3.2).

These results imply that the weights and the dependent variable estimated in a model with one hidden layer are closer to the true DGP values, because of respectively a lower MSE and MSPE, indicating that a model with 2 hidden layers is too complex. These outcomes, including the outcome that an FNN with DeepPINK performs better with one hidden layer compared to an ENN and an EDP with one hidden layer because of lower errors, are also expected, given that the DGP that is formulated is a linear model and, hence, a simpler model would suffice for estimation. The additional information added at each hidden layer with context units does seem to improve the estimations and combining both the knockoff features with the true features seems to lead to a selection of less features, such that the proposed model selects (a subset of) features which may be sufficiently relevant for explaining the output.

When testing on 50 features with 100 observations, one can see that when using 1 hidden layer, that the MSE are higher for most of the cases compared to the model with 30 features for the models with DeepPINK, while the MSPE is lower (Table D.1 compared to Table 3.1 and Table D.3 compared to Table 3.3). For the model without DeepPINK this is the other way around. These results imply that the weights are predicted with a higher error and the outcomes with a lower error when using DeepPINK with 1 hidden layer in a DGP with more variables in the generated dataset, while this is the other way around when not using a DeepPINK filter.

For the models with 2 hidden layers the MSE and MSPE are lower for the models with DeepPINK (Table D.2 compared to Table 3.2 and Table D.4 compared to Table 3.4), while it is higher when looking at the ENN. In particular the ENN does not perform well when using more features and

these results cannot be seen as being reliable. The models with one hidden layer do not select a subset of the 50 features, while the EDP and FNN with DeepPINK select slightly less variables when the modified FDR is being used with 2 hidden layers (Table D.2) at a lower MSPE and MSE than the models with 1 hidden layer and modified FDR (Table D.1). These outcomes may indicate that more complex models are needed for feature selection when more features are added into a dataset and that it may be desired to use more batches and more folds to reduce the error measures.

3.2 Empirical application

Data will be used to do an empirical analysis on the introduced model. The purpose of doing this analysis is to see how the models will perform on real data. The reliability, robustness and adaptability of a neural network is determined by the range, quality source and quantity of the used dataset (Lobbrecht et al., 2005), which also emphasizes the need to use a dataset that is large enough. For an empirical analysis a filtered genotype-phenotype correlation dataset for testing on drug susceptibility used in the studies of Rhee, Gonzales, Rami Kantor, Ravela, and Shafer (2003) and Rhee et al. (2006) is implemented. Drug susceptibility data has been collected from the Human immunodeficiency virus Drug Resistance Database from Stanford University.

The EDP model will be used to test on feature selection and, hence, identifying mutations that are related to drug resistance in HIV-1 protease. As dependent response variable logarithmic values for the drug resistance level to the protease inhibitor LPV will be used. The feature matrix is a matrix of ones and zeros for 43 features and 1550 instances, where a value of one points to a mutation caused by the drug that was adhered to fight the HIV virus. Because of the high mathematical workload it takes to get the outcomes, 2 epochs, 2 batches and 2 folds will be used and an FDR level of 0.20 will be taken.

Table 3.5: MSPE and number of selected features for the dataset

This table gives the statistical results for a study with 2-fold cross-validation on drug susceptibility data retrieved from Human immunodeficiency virus Drug Resistance Database from Stanford University. The dataset consists of 43 independent variables, 1 dependent variable with values for the drug resistance level to the protease inhibitor LPV and 1550 observations. Each training set and test set are generated using 2-fold cross-validation. Each training set is being split into 2 batches and the number of epochs is 2. The outcomes are given for an FNN and an ENN with 1 hidden layer and with 2 hidden layers, where either the modified FDR or the exact FDR is being controlled.

	ENN		EDP		FNN with DeepPINK filter	
	MSPE	Selected	MSPE	Selected	MSPE	Selected
1 hidden layer, modified FDR	2.042	42	1.282	42	1.296	42
2 hidden layers, modified FDR	75449.100*	42	1.091	43	1.313	43
1 hidden layer, exact FDR	1.716	43	1.282	43	1.296	43
2 hidden layers, exact FDR	75449.100*	43	1.091	43	1.313	43

* The test errors for the first fold are large which yields an MSPE of 150886.700. For the second fold the MSPE is 11.508.

The results for all examined models in Table 3.5 show that, when using 1 hidden layer with the modified FDR, less features are selected as relevant features. The ENN does not seem to give reliable results when 2 hidden layers are used regarding the MSPE. A possible explanation could be that using an ENN with 2 hidden layers is too complex to use for these results. Looking at the models with 1 hidden layer, the EDP estimates with the lowest MSPE for this dataset. The addition of a recurrent property and a DeepPINK filter, hence, seems to estimate the outcomes better.

4 Concluding remarks

The objective of this paper was to investigate if feature selection (variable selection) can be improved with the use of an RNN combined with model-X knockoffs and a DeepPINK model framework. Overall, the outcomes in this study show that an FNN with DeepPINK performs better with one hidden layer compared to an ENN and an EDP with one hidden layer. The use of 1 hidden layer seems to lead to less features being selected as a subset and leading to lower MSPE and MSE compared to the case where the models with 2 hidden layers are being used. When more features and observations are used more complex models may be needed for feature selection. The additional information added in an RNN via a context unit does not seem to improve the estimations, but combining both the knockoff features with the true features seems to lead to a selection of less features.

4.1 Future research

Whether the proposed model performs better than other variants depends upon the configuration of the algorithm and conclusions may alter when altering the architecture of the model even to a small extent. The algorithm of NNs are dependent upon the choice of the type of network (e.g. also the number of nodes in each layer), the activation functions in the layers and in the output layer (e.g. a softmax function), the choice of a regularization method for determining the parameters and the method of data representation (Chong et al., 2017). One has to make a trade-off between using too much or too few hidden layers. In this study no bias terms are used within the model which could also have influenced the outcomes and the conclusions. In contrast to the models used, data-driven models do not need the underlying modeling process. Unsupervised learning methods use the aspect of organizing the features, such that unknown patterns, regularities and classifications can be found (Lobbrecht et al., 2005). No desired output is required for these types of learning methods.

The choice of the calculation of the knockoff statistics, choice of the feature importance measures in case no DeepPINK framework is used and initialization of the weights are also of importance when the architecture of the model is made. The weights are not updated using stochastic gradient descent, but standard gradient descent (Sundermeyer et al., 2014). The LSTM could also be combined with the DeepPINK model framework in order to allow for stochastic weights.

In the empirical analysis, when using an ENN without DeepPINK (so when only the knockoffs are taken into an ENN), it seems that the MSPE is large in the first fold when using 2 hidden layers. It seems that using 2 hidden layers is too complex to model the knockoffs. I have only used the created knockoffs of the variables in an ENN model in line with Lu et al. (2018) where they

also only used the knockoffs in an FNN without DeepPINK. These results may alter when using the original variables, which is also a more logical choice and which is left for future research.

Regarding the simulation study, the range of the true DGP coefficients in the simulation could also be alternated as test, as was done in Foygel Barber and J. Candès (2015), where this study only looked at the signal effect of the model by setting some signals to zero. Also, a non-linear DGP can be tested to see if this makes any difference. This study has not varied in FDR levels to see if this makes any difference, because the focus was on testing if the outcomes on feature selection would differ substantially for different models at a fixed FDR level. Future research can also look at varying FDR levels. A cross-validation has been done in this study where future research can also look at another setting for the folds. More reliable outcomes may be obtained when one increases the number of epochs, folds and batches, which may lead to a better convergence of the weights. The initialization of the weight parameters, FDR level, number of batches, batch size, ratio between the test and the training sample, learning rate, epochs, number of hidden layers, number of observations and number of features can be varied, where this study has not varied all settings due to the mathematical workload to run all simulations.

This study has looked at the MSE and MSPE as performance statistics as it is not directly possible to derive p-values from the estimated weights inside an NN. Future research could maybe have a look at using a bootstrap procedure to simulate a distribution of the weights inside an NN by resampling from the observed weights and by deriving a 95% confidence interval as means of testing null hypotheses for the size of the weights.

This paper has looked at the proposed model with the standard backpropagation algorithm. A possible extension could be to investigate the model performance in the case of time series where the weights are trained with the recurrent Newton algorithm (Gençay & Liu, 1997; Werbos, 1990), with the backpropagation through time (BPTT) algorithm (Sundermeyer et al., 2014) or with a real-time recurrent learning algorithm (Yümlü, Gürgen, & Okay, 2005). Dynamic values for the penalty term at the L1-regularization (Sun, 1999) using a dynamic learning rate (Herbrich et al., 1998) could also give more insights. One can also stop the number of times to go through the training set by setting a threshold based on e.g. the change (or lack of change) in the quadratic error, where this study has used a fixed number of batches to update the weights during the training process. Data snooping can also give additional insights into using the proposed model, where the researcher uses statistical inference on empirical data. Further research is desired in order to get more insights into the use of neural networks and its characteristics.

A Configuration of the ENN

The standard architecture of the RNN has three set of weights at each step in the training process (Sutskever, 2013): One set of weights containing the weights which connect the input layer and the hidden layer, one set with weights going from the hidden layer to the output layer and one set with weights from the hidden layer at the previous time point (or previous step) to the current hidden layer. The node in the hidden layer, which is again used at the next step and which has an effect on the hidden layer at the next step, is also called the context unit and this node characterizes the recurrent property of the NN (Lobbrecht et al., 2005). The weights are calculated such that the previous states of the network are taken into account (Koskela, 2003) and the weights are updated during a training process, which is explained in section 2.

The context unit could be processed by an exponential decay function as choice for an activation function, such that the effect of the context unit would decline during the updates of the weights (Koskela, 2003). In a mathematically enhanced neural network the activation would not only have a zero or a one for activation of a node, but there would also be intermediate steps in the range (0,1) regarding the activation like the logistic function (Kuan, 2006), with the requirement that the activation function is bounded (Kuan, 2006). In line with Lu et al. (2018) I have chosen to use the ReLU activation function instead, as this activation function works best as a general approximator.

The choice of a ReLU activation function to process the hidden layer nodes can be motivated by the finding that it is known to provide faster learning inside the network compared to sigmoid functions, while the network performance stays the same or can even be improved (Chong et al., 2017). The choice of a non-linear activation function can be motivated by the fact that a linear activation function does not have as additional advantage that learning is possible because a derivative does not exist for the backpropagation algorithm and, thus, the weights are not updateable when using a linear activation function.

In order to select which nodes die and through which nodes the data will run through, the parameters (weights) will be estimated during the training process which identifies which parameters and variables are needed for the prediction of the output in the test data (Cook & Smalter Hall, 2017; Lobbrecht et al., 2005). Usually the training process could be used to identify which nodes are relevant for further estimation with the use of implicit regularization. Implicit regularization can be defined in this study as adding information to the weights in an NN to prevent overfitting based on the process itself, e.g. using parameter sharing in convolution layers or stochastic gradient descent. In contrast, explicit regularization methods determine more clearly the outcome of the weights using, e.g, weight decay or dropout (Hernández-García & König, 2018).

When estimating the weights problems with overfitting can arise. Overfitting happens when there is a high variance in the parameters which may lead to predictions being outside of the

range of the training data. Overfitting can happen when the model is too complex (has too many hidden layers) (Sun, 1999) and when there are too much free parameters (the degree or number of non-zero weights) to estimate. In the latter case the model will start memorizing the outcomes instead of predicting them (Gradojevic & Yang, 2000). To prevent overfitting, weight decay will be used as explicit regularization technique on the errors of the parameters. This regularization technique works by adding a penalty term to the loss function in order to reduce the size of the weights (Sun, 1999). This way the weights shrink at each training update, such that the calculated parameters during training will be controlled to take on small values (Nabian & Meidani, 2019). On the other hand, one needs to take into account that if the model is not complex enough, this will lead to underfitting (large bias in the output) (Sun, 1999; Gradojevic & Yang, 2000). Over- and underfitting also occur in case the training data is not sufficient in size (Nusrat & Jang, 2018).

The order of the model is based on the number of hidden nodes, which will be the number of states in the model (Elman, 1990). One rule of thumb for the choice of the number of nodes in the hidden layer is that it should be between the number of input nodes and the number of output nodes (Heaton, 2008). However, given that the ReLU activation function is used in the proposed model, this allows for turning off nodes which are not assumed to be of importance into mapping to the output. In line with Lu et al. (2018) the number of variables (number of nodes) in all layers in the proposed model is assumed to be the same as the number of hidden nodes in each of the hidden layers. The output layer will consist of one node.

Regarding the number of hidden layers, having too many hidden layers may lead to problems for the training process like finding local minima and slow convergence of the parameters (Cao et al., 2017). Having zero hidden layers amounts to the use of the model only in case of linear modeling problems. In the case of having one hidden layer, the model can be used while there is continuous mapping from one finite space (e.g. the feature space) to another finite space (e.g. the output space). Having two hidden layers may be useful for smooth mapping (Heaton, 2008). The activation function that is used here is not the smooth variant, which could imply that having one hidden layer would be sufficient for the proposed model. Heaton (2008) highlights that the given reasons in his paper for the choice of 0, 1 or 2 hidden layers are meant for an FNN. For the sake of simplicity, I will also use 1 and 2 hidden layers for the proposed RNN in this study. This means that the proposed RNN in this study will have at least 2 set of weights after the filter: One set of weights from the filtered input to the hidden layer and one from the hidden layer to the output.

B Training the model

Training is done to determine the contribution of each pairwise couple and to calculate the weights in the layers. For the purpose of training the data in order to estimate the weights, the dataset will be split into a training set and a test set (hold-out sample). The training set will also be split into batches, where the weights will be estimated within each batch and updated at the end of each batch. A common approach for choosing the batch size during training is by choosing a fixed size.

Assume that a sample consists of one row (one observation/instance). This implies that the whole dataset consists of N samples in a dataset while the training sample consists of many rows altogether of the dataset (many samples) (Brownlee, 2018). Take $1 < S < N$ samples as the training sample. A batch of size $1 \leq b \leq S$ would then amount to having b samples within one batch and having $\frac{S}{b}$ batches within one training sample. In this case the dataset would have a test set with $N - S$ samples.

For this study I have chosen to use c folds for creating the training set from the N instances, instead of choosing a fixed training sample. For each fold, the original sample will be randomly partitioned into c sub-samples. This means that the algorithm will be going through $c-1$ training sets and 1 test set. All chosen sets of the dataset are used for both training and validation, and each set is used for validation exactly once. As an example, say, that the dataset consists of $N = 100$ instances. Let's say that $c = 10$ folds. I will iterate the number of folds from 1 to 10, making each fold the test set and making the folds that are not c the training set. Then the training set would have $S = N - \frac{N}{c} = 100 - \frac{100}{10} = 90$ instances, where each 10 instances would be taken as test set and the remaining 90 instances are taken as training set. The procedure for selecting the training set and test set can also be seen as a c-fold cross-validation, which is approximately the same approach as in Rhee et al. (2006), who train and test NNs, among other learning methods, in their study with the use of a 5-fold cross-validation.

For the batch size within each training set I will take a ratio of $\frac{1}{\text{number of batches}}$, creating the specified number of batches of size $b = \frac{S}{\text{number of batches}}$. This means that the training will be done for c training sets where each training set will be divided into the specified number of batches, such that a total of $\textit{number of batches} * c$ batches will be created. After each batch of size b the weights will be updated, which means that a batch will be taken into the model to calculate the weights before taking the next batch into the model.

The weight parameters will be calculated within the batches inside the training set. These batches will be gone through for a number of epochs, where the number of epochs is the number of times to go through the entire training set. The number of epochs can be set to an integer value between one and infinity. One can also stop the number of times to go through the training set by setting a threshold based on e.g. the change (or lack of change) in the quadratic error.

This gives $number\ of\ batches * c * epochs$ batches during the whole training process, which is also the total number of updates performed. The update for each batch will be done for a number of epochs, where the algorithm goes over the whole training set for a number of epochs. Repeating the procedure for a number of epochs would on average lead to the desired outcomes. Therefore, in order to determine the classification rule from the test set, the cross-validation will be done repeatedly. After the training process the found parameters (weights) are then used in the test sample to determine the accuracy of deriving the output in the test sample (Sun, 1999).

C Backpropagation

The backpropagation algorithm is used to update the weights during the training process. In this part the algorithm will be derived that is needed for the EDP, the proposed model. First, define the following relation for the derivative of a random scalar weight w:

$$\frac{\partial |w|}{\partial w} = \begin{cases} +1, & \text{if } w > 0 \\ -1, & \text{if } w < 0 \end{cases} \tag{C.1}$$

From this relation one can see that the node has no effect at all when a weight is zero, which means that no derivative has to be calculated when the weight is set to zero because no regularization is possible in this case.

The backpropagation will start with finding the partial derivative of the quadratic error function for each i-th ($i = 1, ...k$ variables) weight between the last hidden layer in the model and the output node (Rumelhart et al., 1986; Pham & Liu, 1996) $w^i_{U+1,t}$ which is:

$$\begin{aligned} \frac{\partial E_t^{L_1}}{\partial w^i_{U+1,t-1}} &= \left[\frac{\partial E_t}{\partial O_t}\right]^T \frac{\partial O_t}{\partial w_{U+1,t-1}} + \lambda \frac{\partial |w^i_{U+1,t-1}|}{\partial w^i_{U+1,t-1}} \\ &= \big[-(y_{train} - O_t)\big]^T \frac{\partial O_t}{\partial w^i_{U+1,t-1}} + \lambda \frac{\partial |w^i_{U+1,t-1}|}{\partial w^i_{U+1,t-1}} \\ &= \big[-(y_{train} - O_t)\big]^T H_t^{U,i} + \lambda \frac{\partial |w^i_{U+1,t-1}|}{\partial w^i_{U+1,t-1}} \\ &= \begin{cases} \big[(O_t - y_{train})\big]^T H_t^{U,i} + \lambda, & \text{if } w^i_{U+1,t-1} > 0 \\ \big[(O_t - y_{train})\big]^T H_t^{U,i} - \lambda, & \text{if } w^i_{U+1,t-1} < 0 \end{cases} \\ &= C'(w^i_{U+1,t-1}). \end{aligned}$$

Here $\square^T$ is the transpose, which is taken for the layer errors that are calculated later. Now, define the following relation:

$$L^{\tilde{q}} = \begin{cases} \prod_{q\in(U-1,U-2,...,\tilde{q})} \left(\sum_i \frac{\partial H^{q+1,i}}{\partial \nu_t^{q+1,i}} \frac{\partial \nu_t^{q+1,i}}{\partial H^{q,i}}\right) = \prod_{q\in(U-1,U-2,...,\tilde{q})} \left(\sum_i f'(\nu_t^{q+1,i}) w^{j,i}_{q+1,t-1}\right), & \text{if } 1 \le \tilde{q} < U \\ 1, & \text{if } \tilde{q} = U \end{cases}$$
$$\text{for } i = 1, ..., k, \tag{C.2}$$

with U the number of hidden layers and i the i-th node corresponding to the i-th feature. The derivative of the cost function with respect to the i-th weight from the 1-st hidden layer,

connecting the j-th node I^j in the input layer of the ENN with the i-th node $\nu^{1,i}$ in the first hidden layer (when $\tilde{q}=1$ in equation (C.2)), can be defined as:

$$\begin{aligned}
\frac{\partial E_t}{\partial w_{1,t-1}^{j,i}} &= \left[\frac{\partial E_t}{\partial O_t}\frac{\partial O_t}{\partial H_t^{U,i}}L^1\frac{\partial H_t^{1,i}}{\partial \nu_t^{1,i}}\right]^T\frac{\partial \nu_t^{1,i}}{\partial w_{1,t-1}^{j,i}}+\lambda\frac{\partial|w_{1,t-1}^{j,i}|}{\partial w_{1,t-1}^{j,i}}\\
&= \left[(O_t-y_{train})w_{U+1,t-1}^i\prod_{q\in(U-1,\dots,1)}\Big(\sum_i f'(\nu_t^{q+1,i})w_{q+1,t-1}^{j,i}\Big)f'(\nu_t^{1,i})\right]^T I_t^j+\lambda\frac{\partial|w_{1,t-1}^{j,i}|}{\partial w_{1,t-1}^{j,i}}\\
&= \begin{cases}\left[(O_t-y_{train})w_{U+1,t-1}^i\prod_{q\in(U-1,\dots,1)}\Big(\sum_i f'(\nu_t^{q+1,i})w_{q+1,t-1}^{j,i}\Big)f'(\nu_t^{1,i})\right]^T I_t^j+\lambda, & \text{if } w_{1,t-1}^{j,i}>0\\ \left[(O_t-y_{train})w_{U+1,t-1}^i\prod_{q\in(U-1,\dots,1)}\Big(\sum_i f'(\nu_t^{q+1,i})w_{q+1,t-1}^{j,i}\Big)f'(\nu_t^{1,i})\right]^T I_t^j-\lambda, & \text{if } w_{1,t-1}^{j,i}<0\end{cases}\\
&= C'(w_{1,t-1}^{j,i}).
\end{aligned}$$

In case $U=1$, there is only 1 hidden layer for which the weight is the weight between the input layer of the ENN model and the first hidden layer. The derivative of the cost function for the U-th hidden layer (when $\tilde{q}=U$ in equation (C.2)) would be:

$$\begin{aligned}
\frac{\partial E_t}{\partial w_{U,t-1}^{j,i}} &= \left[\frac{\partial E_t}{\partial O_t}\frac{\partial O_t}{\partial H_t^{U,i}}L^U\frac{\partial H_t^{U,i}}{\partial \nu_t^{U,i}}\right]^T\frac{\partial \nu_t^{U,i}}{\partial w_{U,t-1}^{j,i}}+\lambda\frac{\partial|w_{U,t-1}^{j,i}|}{\partial w_{U,t-1}^{j,i}}\\
&= \left[(O_t-y_{train})w_{U+1,t-1}^i f'(\nu_t^{U,i})\right]^T H_t^{U-1,j}+\lambda\frac{\partial|w_{U,t-1}^{j,i}|}{\partial w_{U,t-1}^{j,i}}\\
&= \begin{cases}\left[(O_t-y_{train})w_{U+1,t-1}^i f'(\nu_t^{U,i})\right]^T H_t^{U-1,j}+\lambda, & \text{if } w_{U,t-1}^{j,i}>0\\ \left[(O_t-y_{train})w_{U+1,t-1}^i f'(\nu_t^{U,i})\right]^T H_t^{U-1,j}-\lambda, & \text{if } w_{U,t-1}^{j,i}<0\end{cases}\\
&= C'(w_{U,t-1}^{j,i}).
\end{aligned}$$

From the $U-1$-st hidden layer all the way back to the second hidden layer, the derivative of the cost function in the $\tilde{q}$-th hidden layer would amount to:

$$\begin{aligned}
\frac{\partial E_t}{\partial w_{\tilde{q},t-1}^{j,i}} &= \left[\frac{\partial E_t}{\partial O_t}\frac{\partial O_t}{\partial H_t^{U,i}}L^{\tilde{q}}\frac{\partial H_t^{\tilde{q},i}}{\partial \nu_t^{\tilde{q},i}}\right]^T\frac{\partial \nu_t^{\tilde{q},i}}{\partial w_{\tilde{q},t-1}^{j,i}}+\lambda\frac{\partial|w_{\tilde{q},t-1}^{j,i}|}{\partial w_{\tilde{q},t-1}^{j,i}}\\
&= \left[(O_t-y_{train})w_{U+1,t-1}^i\prod_{q\in(U-1,\dots,\tilde{q}))}\Big(\sum_i f'(\nu_t^{q+1,i})w_{q+1,t-1}^{j,i}\Big)f'(\nu_t^{\tilde{q},i})\right]^T H_t^{\tilde{q}-1,j}+\lambda\frac{\partial|w_{\tilde{q},t-1}^{j,i}|}{\partial w_{\tilde{q},t-1}^{j,i}}\\
&= \begin{cases}\left[(O_t-y_{train})w_{U+1,t-1}^i\prod_{q\in(U-1,\dots,\tilde{q})}\Big(\sum_i f'(\nu_t^{q+1,i})w_{q+1,t-1}^{j,i}\Big)f'(\nu_t^{\tilde{q},i})\right]^T H_t^{\tilde{q}-1,j}+\lambda, & \text{if } w_{\tilde{q},t-1}^{j,i}>0\\ \left[(O_t-y_{train})w_{U+1,t-1}^i\prod_{q\in(U-1,\dots,\tilde{q})}\Big(\sum_i f'(\nu_t^{q+1,i})w_{q+1,t-1}^{j,i}\Big)f'(\nu_t^{\tilde{q},i})\right]^T H_t^{\tilde{q}-1,j}-\lambda, & \text{if } w_{\tilde{q},t-1}^{j,i}<0\end{cases}\\
&= C'(w_{\tilde{q},t-1}^{j,i}).
\end{aligned}$$

The derivative of the cost function for the i-th weight for the ENN input layer is derived as follows:

$$
\begin{aligned}
&\frac{\partial E_t}{\partial w^i_{0,t-1}} \\
&= \left[\frac{\partial E_t}{\partial O_t}\frac{\partial O_t}{\partial H^{U,i}_t}L^1\frac{\partial H^{1,i}_t}{\partial \nu^{1,i}_t}\frac{\partial \nu^{1,i}_t}{\partial I^i_t}\frac{\partial I^i_t}{\partial R^i_t}\right]^T \frac{\partial R^i_t}{\partial w^i_{0,t-1}} + \lambda\frac{\partial |w^i_{0,t-1}|}{\partial w^i_{0,t-1}} \\
&= \left[(O_t - y_{train})w^i_{U+1,t-1}\prod_{q\in(U-1,\dots,1)}\Big(\sum_i f'(\nu^{q+1,i}_t)w^{j,i}_{q+1,t-1}\Big)f'(\nu^{1,i}_t)w^{j,i}_{1,t-1}f'(I^i_t)\right]^T G^i_t + \lambda\frac{\partial |w^i_{0,t-1}|}{\partial w^i_{0,t-1}} \\
&= \begin{cases} \left[(O_t - y_{train})w^i_{U+1,t-1}\prod_{q\in(U-1,\dots,1)}\Big(\sum_i f'(\nu^{q+1,i}_t)w^{j,i}_{q+1,t-1}\Big)f'(\nu^{1,i}_t)w^{j,i}_{1,t-1}f'(I^i_t)\right]^T G^i_t + \lambda, & \text{if } w^i_{0,t-1} > 0 \\ \left[(O_t - y_{train})w^i_{U+1,t-1}\prod_{q\in(U-1,\dots,1)}\Big(f'(\nu^{q+1,i}_t)w^{j,i}_{q+1,t-1}\Big)f'(\nu^{1,i}_t)w^{j,i}_{1,t-1}f'(I^i_t)\right]^T G^i_t - \lambda, & \text{if } w^i_{0,t-1} < 0 \end{cases} \\
&= C'(w^i_{0,t-1}).
\end{aligned}
$$

The derivative of the cost function for the i-th filter weight for the pairwise-coupling layer will be determined as follows, where the same derivation is used for $\tilde{z}^i_{t-1}$:

$$
\begin{aligned}
&\frac{\partial E_t}{\partial z^i_{t-1}} \\
&= \left[\frac{\partial E_t}{\partial O_t}\frac{\partial O_t}{\partial H^{U,i}_t}L^1\frac{\partial H^{1,i}_t}{\partial \nu^{1,i}_t}\frac{\partial \nu^{1,i}_t}{\partial I^i_t}\frac{\partial I^i_t}{\partial R^i_t}\frac{\partial R^i_t}{\partial G^i_t}\frac{\partial G^i_t}{\partial F^i_t}\right]^T \frac{\partial F^i_t}{\partial z^i_{t-1}} + \lambda\frac{\partial |z^i_{t-1}|}{\partial z^i_{t-1}} \\
&= \left[(O_t - y_{train})w^i_{U+1,t-1}\prod_{q\in(U-1,\dots,1)}\Big(\sum_i f'(\nu^{q+1,i}_t)w^{j,i}_{q+1,t-1}\Big)f'(\nu^{1,i}_t)w^{j,i}_{1,t-1}f'(I^i_t)w^i_{0,t}f'(F^i_t)\right]^T \mathbf{x}^i + \lambda\frac{\partial |z^i_{t-1}|}{\partial z^i_{t-1}} \\
&= \begin{cases} \left[(O_t - y_{train})w^i_{U+1,t-1}\prod_{q\in(U-1,\dots,1)}\Big(\sum_i f'(\nu^{q+1,i}_t)w^{j,i}_{q+1,t-1}\Big)f'(\nu^{1,i}_t)w^{j,i}_{1,t-1}f'(I^i_t)w^i_{0,t}f'(F^i_t)\right]^T \mathbf{x}^i + \lambda, & \text{if } z^i_{t-1} > 0 \\ \left[(O_t - y_{train})w^i_{U+1,t-1}\prod_{q\in(U-1,\dots,1)}\Big(\sum_i f'(\nu^{q+1,i}_t)w^{j,i}_{q+1,t-1}\Big)f'(\nu^{1,i}_t)w^{j,i}_{1,t-1}f'(I^i_t)w^i_{0,t}f'(F^i_t)\right]^T \mathbf{x}^i - \lambda, & \text{if } z^i_{t-1} < 0 \end{cases} \\
&= C'(z^i_{t-1}).
\end{aligned}
$$

As the derivatives of the cost functions make use of duplicate calculations, as seen from equation (C.2) and from the derivation of the derivatives, these duplicate calculations from the previous layer can be used to calculate the derivatives at the next layers and, thus, to simplify the derivation. In this paper I will apply the formulas for calculating the derivatives $C'(w^i_{U+1,t-1})$, $C'(w^{j,i}_{q,t-1})$, $C'(w^{j,i}_{1,t-1})$ and $C'(w^i_0)$ as follows:

1. First the layer errors without regularization will be calculated as:
   (a) $E_{O,t} = (O_{train,t} - y_{train})$,
   where $E_{O,t-1}$ is the output layer error going back from the output layer to the last hidden layer, y_{train} is the true output in the training set and $O_{train,t}$ is the predicted output for the training set.
   (b) $E_{H^{U,i}_t} = E_{O,t}w^i_{U+1,t-1}f'(\nu^{U,i}_t)$,
   where $E_{H^{U,i}_t}$ is the U-th hidden layer error for the i-th node going back from the U-th hidden layer to the $U-1$-st hidden layer, $w^i_{U+1,t-1}$ is the scalar output weight from

the previous training update which connects the i-th node from the last hidden layer with the output node at $t-1$ and where $f'(\cdot)$ is the derivative of the ReLU activation function. This expression also holds when the number of hidden layers is 1 in the model. In that case steps 1c and 1d do not have to be derived.

(c) $E_{H_t^{\tilde{q},j}} = E_{O,t} w^j_{U+1,t-1} \prod_{q\in(N-1,\dots,\tilde{q})} \Big(\sum_i f'(\nu_t^{q+1,i}) w^{j,i}_{q+1,t-1} \Big) f'(\nu_t^{\tilde{q},j})$
$= \sum_i E_{H_t^{\tilde{q}+1,i}} w^{j,i}_{\tilde{q}+1,t-1} f'(\nu_t^{\tilde{q},j})$,
for $\tilde{q}$ going back from the $U-1$-st hidden layer to the second hidden layer. Here $E_{H_t^{\tilde{q},j}}$ is the hidden layer error going back from the $\tilde{q}$-th hidden layer to the $\tilde{q}-1$-st hidden layer for the i-th node in the $\tilde{q}$-th hidden layer, $\nu_{t-1}^{q+1,i}$ is the i-th hidden node not processed by the ReLU activation function and $w^{j,i}_{q+1,t-1}$ is the weight connecting the j-th node from the last hidden layer with the i-th node of the following hidden layer.

(d) $E_{H_t^{1,j}} = E_{O,t} w^j_{U+1,t-1} \prod_{q\in(N-1,\dots,1)} \Big(\sum_i f'(\nu_t^{q+1,i}) w^{j,i}_{q+1,t-1} \Big) f'(\nu_t^{1,j})$
$= \sum_i E_{H_t^{2,i}} w^{j,i}_{2,t-1} f'(\nu_t^{1,j})$ if the number of hidden layers is at least 2,
where $E_{H_t^{1,j}}$ is the hidden layer error from the j-th node going back in the first hidden layer to the ENN input layer.

(e) $E_{I_t^{1,j}} = E_{O,t} w^j_{U+1,t-1} \prod_{q\in(U-1,\dots,1)} \Big(\sum_i f'(\nu_t^{q+1,i}) w^{j,i}_{q+1,t-1} \Big) f'(\nu_t^{1,i}) w^{j,i}_{1,t-1} f'(I_t^j)$
$= \sum_i E_{H_t^{1,i}} w^{j,i}_{1,t-1} f'(I_t^j)$,
where $E_{I_t^{1,j}}$ is the ENN input layer error going back from the ENN input layer to the pairwise-coupling layer.

(f) $E_{F_t^{1,j}} = E_{O,t} w^j_{U+1,t-1} \prod_{q\in(U-1,\dots,1)} \Big(\sum_i f'(\nu_t^{q+1,i}) w^{j,i}_{q+1,t-1} \Big) f'(\nu_t^{1,i}) w^{j,i}_{1,t-1} f'(I_t^j) w^j_{0,t} f'(F_t^j)$
$= \sum_i E_{H_t^{1,i}} w^{j,i}_{1,t-1} f'(I_t^j) w^j_{0,t} f'(F_t^j)$,
where $E_{F_t^{1,j}}$ is the pairwise-coupling layer error going back from the pairwise-coupling layer to the input layer for the true features and the knockoffs.

2. Then the cost of the derivative of the weights will be calculated by minimizing the loss function (in this paper the quadratic cost function is taken as loss function), which can be seen as the product of the current layer error and the current layer input. The derivative of the regularization part will be included:

(a)
$$C'(w^i_{U+1,t-1}) = \begin{cases} (E_{O,t})^T H_t^{U,i} + \lambda, & \text{if } w^i_{U+1,t-1} > 0 \\ (E_{O,t})^T H_t^{U,i} - \lambda, & \text{if } w^i_{U+1,t-1} < 0 \end{cases},$$
where $C'(w^i_{U+1,t-1})$ is the derivative of the cost function for the output weight which connects the i-th node in the last U-th hidden layer and the output node. When $t=1$ one looks at the change of the updated weight at $t=1$ compared to the initialized weight at $t=0$.

(b)
$$C'(w^{j,i}_{U,t-1}) = \begin{cases} (E_{H_t^{U,i}})^T H_t^{U-1,j} + \lambda, & \text{if } w^{j,i}_{U,t-1} > 0 \\ (E_{H_t^{U,i}})^T H_t^{U-1,j} - \lambda, & \text{if } w^{j,i}_{U,t-1} < 0 \end{cases},$$

where $C'(w^{j,i}_{U,t-1})$ is the derivative of the cost function for the last hidden layer which connects the j-th node in the $U-1$-st hidden layer with the i-th node in the U-th hidden layer.

(c)
$$C'(w^{j,i}_{\tilde{q},t-1}) = \begin{cases} (E_{H^{\tilde{q},i}_t})^T H^{\tilde{q}-1,j}_t + \lambda, & \text{if } w^{j,i}_{\tilde{q},t-1} > 0 \\ (E_{H^{\tilde{q},i}_t})^T H^{\tilde{q}-1,j}_t - \lambda, & \text{if } w^{j,i}_{\tilde{q},t-1} < 0 \end{cases},$$

where $C'(w^{j,i}_{\tilde{q},t-1})$ is the derivative of the cost function for the weights in the second hidden layer until the last hidden layer. This step does not need to be made when the number of hidden layers in the model is 1.

(d)
$$C'(w^{j,i}_{1,t-1}) = \begin{cases} (E_{H^{1,i}_t})^T I^j_t + \lambda, & \text{if } w^{j,i}_{1,t-1} > 0 \\ (E_{H^{1,i}_t})^T I^j_t - \lambda, & \text{if } w^{j,i}_{1,t-1} < 0 \end{cases},$$

where $C'(w_{1,t-1})$ is the derivative of the cost function for the weights connecting the ENN input layer with the first hidden layer at step $t-1$.

(e)
$$C'(w^i_{0,t-1}) = \begin{cases} (E_{I^{1,i}_t})^T G^i_t + \lambda, & \text{if } w^i_{0,t-1} > 0 \\ (E_{I^{1,i}_t})^T G^i_t - \lambda, & \text{if } w^i_{0,t-1} < 0 \end{cases},$$

where $C'(w_0)$ is the derivative of the cost function for the weights connecting the pairwise-coupling layer with the ENN input layer.

(f)
$$C'(z^i_{t-1}) = \begin{cases} (E_{F^{1,i}_t})^T \mathbf{x}^i + \lambda, & \text{if } z^i_{t-1} > 0 \\ (E_{F^{1,i}_t})^T \mathbf{x}^i - \lambda, & \text{if } z^i_{t-1} < 0 \end{cases},$$

where $C'(z)$ is the derivative of the cost function for the filter weights connecting the ENN input layer with the true features. The derivation goes in the same way for $\tilde{z}^i_{t-1}$ where $\tilde{\mathbf{x}}^i$ is used insetad of $\mathbf{x}^i$. Regarding 2d, 2e and 2f, the layer error in 1b must be used instead of the layer error in 1d, when the number of hidden layers is 1.

The weights will be updated as (Gómez-Ramos & Venegas-Martínez, 2013; Pham & Liu, 1996; Nabian & Meidani, 2019):

1. $w^i_{U+1,t} = w^i_{U+1,t-1} - lrate * C'(w^i_{U+1,t-1})$,
where $w^i_{U+1,t-1}$ is the updated output weight connecting the i-th node in the U-th hidden layer and the output node and where $lrate$ is the user-specified learning rate. For this algorithm a learning rate of 0.001 will be used in updating the weights in line with Lu et al. (2018).

2. $w^{j,i}_{\tilde{q},t} = w^{j,i}_{\tilde{q},t-1} - lrate * C'(w^{j,i}_{\tilde{q},t-1})$,
where $w^{j,i}_{\tilde{q},t}$ is the updated weight in the $\tilde{q}$-th hidden layer connecting the j-th node in the $\tilde{q}$-th hidden layer with the i-th node in the $\tilde{q}+1$-st hidden layer.

3. $w_{1,t}^{j,i} = w_{1,t-1}^{j,i} - lrate * C'(w_{1,t-1}^{j,i})$,
   where $w_{1,t}^{j,i}$ is the updated weight connecting the j-th node in the ENN input layer with the i-th node in the first hidden layer.

4. $w_{0,t}^{i} = w_{0,t-1}^{i} - lrate * C'(w_{0,t-1}^{i})$,
   where $w_{0,t}^{i}$ is the updated weight connecting the pairwise-coupling layer with the ENN input layer.

5. $z_t^i = z_{t-1}^i - lrate * C'(z_{t-1}^i)$,
   where z_t^i is the updated weight connecting $\mathbf{x}^i$ with the pairwise-coupling layer, where again the same derivation holds for $\tilde{z}_t^i$.

D Simulation on 50 features

Table D.1: Simulation results for one hidden layer and the modified FDR for 50 features

This table gives the statistical results for a simulation study with a 2-fold cross-validation. The number of features that are taken is 50 with 5 sparsity levels. Each training set and test set are generated using 2-fold cross-validation. The simulation is done 2 times, where each training set is being split into 2 batches and where the number of epochs is 2. The outcomes are given for an FNN and an ENN with 1 hidden layer, where the modified FDR is being controlled.															
ENN, 1 hidden layer, modified FDR is controlled															
	MSPE					MSE					Number of selected features				
feature correlation	non-zero signals					non-zero signals					non-zero signals				
	10	20	30	40	50	10	20	30	40	50	10	20	30	40	50
0	42.737	58.329	56.062	52.414	73.227	0.999	0.835	0.838	0.794	0.824	50	50	50	50	50
0.1	30.589	36.596	42.919	47.763	56.702	0.809	0.786	0.797	0.803	0.784	50	50	50	50	50
0.2	64.922	62.145	62.044	66.721	92.339	0.920	0.844	0.827	0.849	0.883	50	50	50	50	50
0.3	46.163	48.569	53.789	71.742	84.151	0.833	0.805	0.817	0.872	0.882	50	50	50	50	50
0.4	52.394	48.464	51.110	57.962	73.500	0.865	0.806	0.811	0.819	0.842	50	50	50	50	50
0.5	38.136	47.321	42.177	51.593	57.559	0.820	0.812	0.804	0.812	0.795	50	50	50	50	50
0.6	61.130	55.455	52.881	58.364	81.424	0.881	0.827	0.812	0.816	0.851	50	50	50	50	50
0.7	62.781	66.099	52.932	69.803	89.585	0.869	0.855	0.793	0.838	0.880	50	50	50	50	50
0.8	22.661	31.900	35.845	42.833	49.926	0.782	0.781	0.792	0.795	0.775	50	50	50	50	50
0.9	26.211	33.686	34.865	44.467	50.653	0.790	0.785	0.788	0.800	0.774	50	50	50	50	50
EDP, 1 hidden layer, modified FDR															
0	9.812	18.180	24.286	32.264	35.766	0.834	0.829	0.830	0.829	0.830	50	50	50	50	50
0.1	8.538	18.295	24.245	32.372	35.726	0.830	0.829	0.830	0.829	0.829	50	50	50	50	50
0.2	8.514	18.179	24.104	32.190	35.472	0.830	0.829	0.830	0.829	0.830	50	50	50	50	50
0.3	8.456	18.135	24.129	32.196	35.647	0.830	0.829	0.830	0.829	0.829	50	50	50	50	50
0.4	8.560	18.349	24.375	32.426	35.699	0.830	0.830	0.830	0.829	0.830	50	50	50	50	50
0.5	8.498	18.217	24.225	32.305	35.795	0.830	0.829	0.830	0.829	0.829	50	50	50	50	50
0.6	8.552	18.361	24.178	32.417	35.695	0.830	0.830	0.830	0.829	0.830	50	50	50	50	50
0.7	8.579	18.317	24.199	32.334	35.747	0.829	0.830	0.830	0.829	0.830	50	50	50	50	50
0.8	8.462	18.282	24.338	32.222	35.770	0.830	0.829	0.830	0.829	0.829	50	50	50	50	50
0.9	8.508	18.363	24.257	32.374	35.801	0.830	0.829	0.830	0.829	0.830	50	50	50	50	50
FNN with DeepPINK filter, 1 hidden layer, modified FDR															
0	9.828	18.081	24.277	32.279	35.855	0.834	0.829	0.830	0.830	0.830	50	50	50	50	50
0.1	8.542	18.178	24.211	32.335	35.815	0.829	0.829	0.829	0.829	0.829	50	50	50	50	50
0.2	8.511	18.106	24.112	32.215	35.571	0.829	0.829	0.830	0.830	0.830	50	50	50	50	50
0.3	8.457	18.024	24.123	32.195	35.756	0.829	0.829	0.830	0.829	0.829	50	50	50	50	50
0.4	8.556	18.262	24.371	32.421	35.770	0.829	0.829	0.830	0.830	0.830	50	50	50	50	50
0.5	8.505	18.146	24.216	32.297	35.850	0.829	0.829	0.829	0.829	0.829	50	50	50	50	50
0.6	8.550	18.308	24.192	32.434	35.743	0.829	0.830	0.830	0.830	0.830	50	50	50	50	50
0.7	8.571	18.226	24.178	32.359	35.811	0.829	0.829	0.829	0.830	0.830	50	50	50	50	50
0.8	8.465	18.194	24.310	32.250	35.870	0.829	0.829	0.829	0.829	0.829	50	50	50	50	50
0.9	8.517	18.269	24.239	32.439	35.939	0.829	0.829	0.829	0.829	0.829	50	50	50	50	50

Table D.2: Simulation results for two hidden layers and the modified FDR for 50 features

This table gives the statistical results for a simulation study with a 2-fold cross-validation.
The number of features that are taken is 50 with 5 sparsity levels.
Each training set and test set are generated using 2-fold cross-validation.
The simulation is done 2 times, where each training set
is being split into 2 batches and where the number of epochs is 2.
The outcomes are given for an FNN and an ENN with 2 hidden layers,
where the modified FDR is being controlled.

ENN, 2 hidden layers, modified FDR is controlled															
	MSPE					MSE					Number of selected features				
feature correlation	non-zero signals					non-zero signals					non-zero signals				
	10	20	30	40	50	10	20	30	40	50	10	20	30	40	50
0	12748.048	7785.567	19817.368	8567.161	8904.998	11.179	6.454	29.329	8.719	10.633	50	50	50	50	50
0.1	9050.837	11047.582	11763.750	8816.723	11936.829	7.058	8.438	12.301	8.761	10.057	50	50	50	50	50
0.2	9035.429	11386.579	6974.194	7480.472	7828.812	7.007	7.890	7.097	6.951	7.034	50	50	50	50	50
0.3	5325.572	4973.869	6183.011	7666.736	5374.245	6.803	5.231	5.895	11.921	6.131	50	50	50	50	50
0.4	5961.887	4974.074	6383.218	6315.341	7489.379	6.393	5.655	6.197	6.181	7.163	49	50	50	50	50
0.5	9496.756	11004.166	17846.911	7581.465	13738.172	7.016	8.291	17.377	6.218	9.226	50	50	50	50	50
0.6	3902.138	3660.489	3697.651	5609.213	5932.143	5.483	4.860	4.717	5.868	6.759	50	50	50	50	50
0.7	5832.897	5321.545	3709.758	12557.563	11488.628	6.091	5.405	5.411	15.768	11.497	50	50	50	50	50
0.8	5700.071	4656.630	6901.040	3966.379	3873.061	6.059	6.089	6.261	4.740	5.562	50	50	50	50	50
0.9	16073.362	5861.295	19589.736	8961.790	7024.310	12.554	6.719	52.755	8.410	7.885	50	50	50	50	50
EDP, 2 hidden layers, modified FDR															
0	10.586	16.862	22.029	30.065	40.446	0.841	0.633	0.632	0.634	0.631	49	49	48	49	48
0.1	7.177	16.446	21.920	29.754	39.996	0.633	0.633	0.632	0.634	0.631	49	49	48	49	48
0.2	7.400	16.559	21.973	29.771	40.214	0.633	0.633	0.632	0.634	0.631	49	49	48	49	48
0.3	7.840	17.684	22.239	30.384	40.588	0.633	0.633	0.632	0.634	0.631	49	49	48	49	48
0.4	7.410	16.550	22.090	30.029	40.372	0.633	0.633	0.632	0.634	0.631	49	49	48	49	48
0.5	7.589	17.219	21.975	30.317	40.598	0.633	0.633	0.632	0.634	0.631	49	49	48	49	48
0.6	7.270	16.638	22.132	30.205	40.442	0.633	0.633	0.632	0.634	0.631	49	49	48	49	48
0.7	7.474	17.227	22.277	30.379	40.608	0.633	0.633	0.632	0.634	0.631	49	49	48	49	48
0.8	7.569	16.934	21.931	30.406	40.137	0.633	0.633	0.632	0.634	0.631	49	49	48	49	48
0.9	7.519	17.261	22.131	30.130	40.235	0.633	0.633	0.632	0.634	0.631	49	49	48	49	48
FNN with DeepPINK filter, 2 hidden layers, modified FDR															
0	9.760	15.819	21.953	30.037	39.881	0.840	0.640	0.640	0.640	0.640	49	49	49	49	49
0.1	6.823	15.778	22.083	30.105	39.936	0.640	0.640	0.640	0.640	0.640	49	49	49	49	49
0.2	6.847	15.761	22.024	30.056	39.949	0.640	0.640	0.640	0.640	0.640	49	49	49	49	49
0.3	6.868	15.822	22.092	30.213	39.946	0.640	0.640	0.640	0.640	0.640	49	49	49	49	49
0.4	6.809	15.772	22.067	30.187	40.054	0.640	0.640	0.640	0.640	0.640	49	49	49	49	49
0.5	6.796	15.773	21.980	30.172	39.969	0.640	0.640	0.640	0.640	0.640	49	49	49	49	49
0.6	6.798	15.740	22.040	30.162	40.022	0.640	0.640	0.640	0.640	0.640	49	49	49	49	49
0.7	6.827	15.752	21.984	30.207	40.012	0.640	0.640	0.640	0.640	0.640	49	49	49	49	49
0.8	6.842	15.759	21.951	30.109	39.972	0.640	0.640	0.640	0.640	0.640	49	49	49	49	49
0.9	6.806	15.739	21.971	30.023	39.847	0.640	0.640	0.640	0.640	0.640	49	49	49	49	49

Table D.3: Simulation results for one hidden layers and the exact FDR for 50 features

This table gives the statistical results for a simulation study with a 2-fold cross-validation.
The number of features that are taken is 50 with 5 sparsity levels.
Each training set and test set are generated using 2-fold cross-validation.
The simulation is done 2 times, where each training set
is being split into 2 batches and where the number of epochs is 2.
The outcomes are given for an FNN and an ENN with 1 hidden layer,
where the exact FDR is being controlled.

ENN, 1 hidden layer, exact FDR is controlled															
	MSPE					MSE					Number of selected features				
feature correlation	non-zero signals					non-zero signals					non-zero signals				
	10	20	30	40	50	10	20	30	40	50	10	20	30	40	50
0	42.737	58.329	56.062	52.414	73.227	0.999	0.835	0.838	0.794	0.824	50	50	50	50	50
0.1	30.589	36.596	42.919	47.763	56.702	0.809	0.786	0.797	0.803	0.784	50	50	50	50	50
0.2	64.922	62.145	62.044	66.721	92.339	0.920	0.844	0.827	0.849	0.883	50	50	50	50	50
0.3	46.163	48.569	53.789	71.742	84.151	0.833	0.805	0.817	0.872	0.882	50	50	50	50	50
0.4	52.394	48.464	51.110	57.962	73.500	0.865	0.806	0.811	0.819	0.842	50	50	50	50	50
0.5	38.136	47.321	42.177	51.593	57.559	0.820	0.812	0.804	0.812	0.795	50	50	50	50	50
0.6	61.130	55.455	52.881	58.364	81.424	0.881	0.827	0.812	0.816	0.851	50	50	50	50	50
0.7	62.781	66.099	52.932	69.803	89.585	0.869	0.855	0.793	0.838	0.880	50	50	50	50	50
0.8	22.661	31.900	35.845	42.833	49.926	0.782	0.781	0.792	0.795	0.775	50	50	50	50	50
0.9	26.211	33.686	34.865	44.467	50.653	0.790	0.785	0.788	0.800	0.774	50	50	50	50	50
EDP, 1 hidden layer, exact FDR															
0	9.812	18.180	24.286	32.264	35.766	0.834	0.829	0.830	0.829	0.830	50	50	50	50	50
0.1	8.538	18.295	24.245	32.372	35.726	0.830	0.829	0.830	0.829	0.829	50	50	50	50	50
0.2	8.514	18.179	24.104	32.190	35.472	0.830	0.829	0.830	0.829	0.830	50	50	50	50	50
0.3	8.456	18.135	24.129	32.196	35.647	0.830	0.829	0.830	0.829	0.829	50	50	50	50	50
0.4	8.560	18.349	24.375	32.426	35.699	0.830	0.830	0.830	0.829	0.830	50	50	50	50	50
0.5	8.498	18.217	24.225	32.305	35.795	0.830	0.829	0.830	0.829	0.829	50	50	50	50	50
0.6	8.552	18.361	24.178	32.417	35.695	0.830	0.830	0.830	0.829	0.830	50	50	50	50	50
0.7	8.579	18.317	24.199	32.334	35.747	0.829	0.830	0.830	0.829	0.830	50	50	50	50	50
0.8	8.462	18.282	24.338	32.222	35.770	0.830	0.829	0.830	0.829	0.829	50	50	50	50	50
0.9	8.508	18.363	24.257	32.374	35.801	0.830	0.829	0.830	0.829	0.830	50	50	50	50	50
FNN with DeepPINK filter, 1 hidden layer, exact FDR															
0	9.828	18.081	24.277	32.279	35.855	0.834	0.829	0.830	0.830	0.830	50	50	50	50	50
0.1	8.542	18.178	24.211	32.335	35.815	0.829	0.829	0.829	0.829	0.829	50	50	50	50	50
0.2	8.511	18.106	24.112	32.215	35.571	0.829	0.829	0.830	0.830	0.830	50	50	50	50	50
0.3	8.457	18.024	24.123	32.195	35.756	0.829	0.829	0.830	0.829	0.829	50	50	50	50	50
0.4	8.556	18.262	24.371	32.421	35.770	0.829	0.829	0.830	0.830	0.830	50	50	50	50	50
0.5	8.505	18.146	24.216	32.297	35.850	0.829	0.829	0.829	0.829	0.829	50	50	50	50	50
0.6	8.550	18.308	24.192	32.434	35.743	0.829	0.830	0.830	0.830	0.830	50	50	50	50	50
0.7	8.571	18.226	24.178	32.359	35.811	0.829	0.829	0.829	0.830	0.830	50	50	50	50	50
0.8	8.465	18.194	24.310	32.250	35.870	0.829	0.829	0.829	0.829	0.829	50	50	50	50	50
0.9	8.517	18.269	24.239	32.439	35.939	0.829	0.829	0.829	0.829	0.829	50	50	50	50	50

Table D.4: Simulation results for two hidden layers and the exact FDR for 50 features

This table gives the statistical results for a simulation study with a 2-fold cross-validation.
The number of features that are taken is 50 with 5 sparsity levels.
Each training set and test set are generated using 2-fold cross-validation.
The simulation is done 2 times, where each training set
is being split into 2 batches and where the number of epochs is 2.
The outcomes are given for an FNN and an ENN with 2 hidden layers,
where the exact FDR is being controlled.

ENN, 2 hidden layers, exact FDR is controlled															
	MSPE					MSE					Number of selected features				
feature correlation	non-zero signals					non-zero signals					non-zero signals				
	10	20	30	40	50	10	20	30	40	50	10	20	30	40	50
0	12748.048	7785.567	19817.368	8567.161	8904.998	11.179	6.454	29.329	8.719	10.633	50	50	50	50	50
0.1	9050.837	11047.582	11763.750	8816.723	11936.829	7.058	8.438	12.301	8.761	10.057	50	50	50	50	50
0.2	9035.429	11386.579	6974.194	7480.472	7828.812	7.007	7.890	7.097	6.951	7.034	50	50	50	50	50
0.3	5325.572	4973.869	6183.011	7666.736	5374.245	6.803	5.231	5.895	11.921	6.131	50	50	50	50	50
0.4	5961.887	4974.074	6383.218	6315.341	7489.379	6.393	5.655	6.197	6.181	7.163	50	50	50	50	50
0.5	9496.756	11004.166	17846.911	7581.465	13738.172	7.016	8.291	17.377	6.218	9.226	50	50	50	50	50
0.6	3902.138	3660.489	3697.651	5609.213	5932.143	5.483	4.860	4.717	5.868	6.759	50	50	50	50	50
0.7	5832.897	5321.545	3709.758	12557.563	11488.628	6.091	5.405	5.411	15.768	11.497	50	50	50	50	50
0.8	5700.071	4656.630	6901.040	3966.379	3873.061	6.059	6.089	6.261	4.740	5.562	50	50	50	50	50
0.9	16073.362	5861.295	19589.736	8961.790	7024.310	12.554	6.719	52.755	8.410	7.885	50	50	50	50	50
EDP, 2 hidden layers, exact FDR															
0	10.586	16.862	22.029	30.065	40.446	0.841	0.633	0.632	0.634	0.631	50	50	50	50	50
0.1	7.177	16.446	21.920	29.754	39.996	0.633	0.633	0.632	0.634	0.631	50	50	50	50	50
0.2	7.400	16.559	21.973	29.771	40.214	0.633	0.633	0.632	0.634	0.631	50	50	50	50	50
0.3	7.840	17.684	22.239	30.384	40.588	0.633	0.633	0.632	0.634	0.631	50	50	50	50	50
0.4	7.410	16.550	22.090	30.029	40.372	0.633	0.633	0.632	0.634	0.631	50	50	50	50	50
0.5	7.589	17.219	21.975	30.317	40.598	0.633	0.633	0.632	0.634	0.631	50	50	50	50	50
0.6	7.270	16.638	22.132	30.205	40.442	0.633	0.633	0.632	0.634	0.631	50	50	50	50	50
0.7	7.474	17.227	22.277	30.379	40.608	0.633	0.633	0.632	0.634	0.631	50	50	50	50	50
0.8	7.569	16.934	21.931	30.406	40.137	0.633	0.633	0.632	0.634	0.631	50	50	50	50	50
0.9	7.519	17.261	22.131	30.130	40.235	0.633	0.633	0.632	0.634	0.631	50	50	50	50	50
FNN with DeepPINK filter, 2 hidden layers, exact FDR															
0	9.760	15.819	21.953	30.037	39.881	0.840	0.640	0.640	0.640	0.640	50	50	50	50	50
0.1	6.823	15.778	22.083	30.105	39.936	0.640	0.640	0.640	0.640	0.640	50	50	50	50	50
0.2	6.847	15.761	22.024	30.056	39.949	0.640	0.640	0.640	0.640	0.640	50	50	50	50	50
0.3	6.868	15.822	22.092	30.213	39.946	0.640	0.640	0.640	0.640	0.640	50	50	50	50	50
0.4	6.809	15.772	22.067	30.187	40.054	0.640	0.640	0.640	0.640	0.640	50	50	50	50	50
0.5	6.796	15.773	21.980	30.172	39.969	0.640	0.640	0.640	0.640	0.640	50	50	50	50	50
0.6	6.798	15.740	22.040	30.162	40.022	0.640	0.640	0.640	0.640	0.640	50	50	50	50	50
0.7	6.827	15.752	21.984	30.207	40.012	0.640	0.640	0.640	0.640	0.640	50	50	50	50	50
0.8	6.842	15.759	21.951	30.109	39.972	0.640	0.640	0.640	0.640	0.640	50	50	50	50	50
0.9	6.806	15.739	21.971	30.023	39.847	0.640	0.640	0.640	0.640	0.640	50	50	50	50	50

References

Arnerić, J., Šestanović, T., & Aljinović, Z. (2014, 12). GARCH based artificial neural networks in forecasting conditional variance of stock returns. *Croatian Operational Research Review*, *5*, 329–343. doi: 10.17535/crorr.2014.0017

Bates, S., Candès, E., Janson, L., & Wang, W. (2019, 03). *Metropolized Knockoff Sampling.*

Benjamini, Y., & Hochberg, Y. (1995). Controlling the False Discovery Rate: A Practical and Powerful Approach to Multiple Testing. *Journal of the Royal Statistical Society. Series B (Methodological)*, *57*(1), 289–300.

Binner, J., Elger, T., Nilsson, B., & Tepper, J. (2005, 03). Tools for non-linear time series forecasting in economics - An empirical comparison of regime switching vector autoregressive models and recurrent neural networks. *Advances in Econometrics*, *19*. doi: 10.1016/S0731-9053(04)19003-8

Brownlee, J. (2018, July). *Difference Between a Batch and an Epoch in a Neural Network.* Retrieved 03–10–2019, from https://machinelearningmastery.com/difference-between-a-batch-and-an-epoch/

Candès, E., Fan, Y., Janson, L., & Lv, J. (2018, 10). Panning for Gold: Model-free Knockoffs for High-dimensional Controlled Variable Selection. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, *80*, 551-–577. (Part 3)

Cao, W., Wang, X.-Z., Ming, Z., & Gao, J. (2017, 09). A Review on Neural Networks with Random Weights. *Neurocomputing*, *275*. doi: 10.1016/j.neucom.2017.08.040

Carlson, D. (1986). What are Schur complements, anyway? *Linear Algebra and its Applications*, *74*, 257 – 275. Retrieved from http://www.sciencedirect.com/science/article/pii/0024379586901278 doi: https://doi.org/10.1016/0024-3795(86)90127-8

Chandrasekhar, A. (2016). Econometrics of Network Formation..

Chappelier, J.-C., Gori, M., & Grumbach, A. (2001). Time in Connectionist Models. In *Sequence Learning - Paradigms, Algorithms, and Applications* (pp. 105–134). London, UK, UK: Springer-Verlag.

Cheng, B., & Titterington, D. M. (1994). Neural Networks: A Review from a Statistical Perspective. *Statistical Science*, *9*(1), 2–30.

Chong, E., Han, C., & Park, F. (2017, 04). Deep Learning Networks for Stock Market Analysis and Prediction: Methodology, Data Representations, and Case Studies. *Expert Systems with Applications*, *83*. doi: 10.1016/j.eswa.2017.04.030

Cook, T. R., & Smalter Hall, A. (2017, September). *Macroeconomic Indicator Forecasting with Deep Neural Networks* (Research Working Paper No. RWP 17-11). Federal Reserve Bank of Kansas City. Retrieved from https://ideas.repec.org/p/fip/fedkrw/rwp17-11.html

de Paula, A. (2016). *Econometrics of network models* (CeMMAP working papers No. CWP06/16). University College London, São Paulo School of Economics: Centre for Microdata Methods and Practice, Institute for Fiscal Studies.

Elman, J. L. (1990). Finding structure in time. *COGNITIVE SCIENCE*, *14*(2), 179–211.

Foygel Barber, R., & J. Candès, E. (2014). *Controlling the false discovery rate via knockoffsy.* (Working paper, arXiv:1809.01185)

Foygel Barber, R., & J. Candès, E. (2015, 03). Controlling the False Discovery Rate via Knockoffs. *The Annals of Statistics*, *43*(5), 2055–2085. doi: 10.1214/15-AOS1337

Foygel Barber, R., & J. Candès, E. (2016, 02). *A knockoff filter for high-dimensional selective inference.*

Foygel Barber, R., J. Candès, E., & J. Samworth, R. (2019, 02). *Robust inference with knockoffs.* (arXiv:1801.03896v4)

Foygel Barber, R., & Ramdas, A. (2016, 11). The p-filter: Multilayer false discovery rate control for grouped hypotheses. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, *79*. doi: 10.1111/rssb.12218

Franses, P. H., & Draisma, G. (1997). Recognizing changing seasonal patterns using artificial neural networks. *Journal of Econometrics*, *81*(1), 273 - 280. doi: https://doi.org/10.1016/S0304-4076(97)00047-X

Gao, T., & Jojic, V. (2016). Degrees of Freedom in Deep Neural Networks. In *Proceedings of the Thirty-Second Conference on Uncertainty in Artificial Intelligence* (pp. 232–241). Arlington, Virginia, United States: AUAI Press.

Gençay, R., & Liu, T. (1997, September). Nonlinear Modelling and Prediction with Feedforward and Recurrent Networks. *Phys. D*, *108*(1-2), 119–134. Retrieved from http://dx.doi.org/10.1016/S0167-2789(97)82009-X doi: 10.1016/S0167-2789(97)82009-X

Gradojevic, N., & Yang, J. (2000, 01). The Application of Artificial Neural Networks to Exchange Rate Forecasting: The Role of Market Microstructure Variables. *Bank of Canada, Working Papers*.

Gómez-Ramos, E., & Venegas-Martínez, F. (2013). A Review of Artificial Neural Networks: How Well Do They Perform in Forecasting Time Series? *Analítika*, *6*(2), 7–15.

Heaton, J. (2008). *Introduction to Neural Networks for Java.* (isbn: 1604390085)

Herbrich, R., Keilbach, M., Graepel, T., Bollmann-sdorra, P., & Obermayer, K. (1998, 08). Neural Networks in Economics: Background, Applications and New Developments. *Advances in Computational Economics*, *11*.

Hernández-García, A., & König, P. (2018). Data augmentation instead of explicit regularization. *CoRR*, *abs/1806.03852*. Retrieved from http://arxiv.org/abs/1806.03852

Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, *9*(8), 1735–1780.

Jaeger, H., & Haas, H. (2004, 05). Harnessing Nonlinearity: Predicting Chaotic Systems and Saving Energy in Wireless Communication. *Science (New York, N.Y.)*, *304*, 78–80. doi: 10.1126/science.1091277

Jianming, Y. (1998). On Measuring and Correcting the Effects of Data Mining and Model Selection. *Journal of the American Statistical Association*, *93*(441), 120–131.

Jordan, M. I. (1986). *Serial Order: A Parallel, Distributed Processing Approach* (Tech. Rep. No. 8604). San Diego: Institute for Cognitive Science, University of California.

Jordan, M. I. (1997). Chapter 25 - serial order: A parallel distributed processing approach. In J. W. Donahoe & V. P. Dorsel (Eds.), *Neural-network models of cognition* (Vol. 121, pp. 471 – 495). North-Holland. Retrieved from http://www.sciencedirect.com/science/article/pii/S0166411597801112 doi: https://doi.org/10.1016/S0166-4115(97)80111-2

Katsevich, E., & Sabattiy, C. (2018). *Multilayer Knockoff Filter: Controlled variable selection at multiple resolutions.* (arXiv:1706.09375)

Koskela, T. (2003). *Neural network methods in analysing and modelling time varying processes* (Unpublished doctoral dissertation). Helsinki University of Technology.

Kuan, C.-M. (2006). *Artificial Neural Networks.*

Kuan, C.-M., & Liu, T. (1995). Forecasting Exchange Rates Using Feedforward and Recurrent Neural Networks. *Journal of Applied Econometrics*, *10*(4), 347–64.

Lobbrecht, A., Dibike, Y., & Solomatine, D. (2005, 03). Neural Networks and Fuzzy Systems in Model Based Control of the Overwaard Polder. *Journal of Water Resources Planning and Management-ASCE*, *131*. doi: 10.1061/(ASCE)0733-9496(2005)131:2(135)

Lu, Y., Fan, Y., Lv, J., & Stafford Noble, W. (2018). Deeppink: reproducible feature selection in deep neural networks. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, & R. Garnett (Eds.), *Advances in Neural Information Processing Systems 31* (pp. 8676–8686). Curran Associates, Inc.

Nabian, M. A., & Meidani, H. (2019, Sep). Physics-Driven Regularization of Deep Neural Networks for Enhanced Engineering Design and Analysis. *Journal of Computing and Information Science in Engineering*, *20*(1). Retrieved from http://dx.doi.org/10.1115/1.4044507 doi: 10.1115/1.4044507

Nagpal, A. (2017, October). *L1 and L2 Regularization Methods.* Retrieved 16–11–2019, from https://towardsdatascience.com/l1-and-l2-regularization-methods-ce25e7fc831c

Nusrat, I., & Jang, S.-B. (2018, 11). A Comparison of Regularization Techniques in Deep Neural Networks. *Symmetry*, *10*, 648. doi: 10.3390/sym10110648

Pham, D. T., & Liu, X. (1996). Training of Elman networks and dynamic system modelling. *International Journal of Systems Science*, *27*(2), 221–226.

R Core Team. (2019). R: A Language and Environment for Statistical Computing [Computer software manual]. Vienna, Austria. Retrieved from http://www.R-project.org/

Rhee, S.-Y., Gonzales, M. J., Rami Kantor, B. J. B., Ravela, J., & Shafer, R. W. (2003). Human Immunodeficiency Virus Reverse Transcriptase and Protease Sequence Database. *Nucleic Acids Research*, *31*(1), 298—303. doi: 10.1093/nar/gkg100

Rhee, S.-Y., Taylor, J., Wadhera, G., Ben-Hur, A., Brutlag, D. L., & Shafer, R. W. (2006, 11). Genotypic predictors of human immunodeficiency virus type 1 drug resistance. *PNAS*, *103*(46), 17355–17360. Retrieved from https://doi.org/10.1073/pnas.0607274103

Rohwer, R. (1990). NEURAL NETWORKS FOR TIME-VARYING DATA. University of Edinburgh 80 South Bridge. (Survey presentation for the workshop: Neural Networks for Statistical and Economic Data Dublin, 10-11 December 1990)

Romano, Y., Sesia, M., & J. Candès, E. (2018, 11). *Deep Knockoffs.* (arXiv:1811.06687)

Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1986, October). Learning representations by back-propagating errors. *Nature*, *323*, 533–536. Retrieved from http://dx.doi.org/10.1038/323533a0

Sesia, M., Katsevich, E., Bates, S., Candès, E., & Sabatti, C. (2019). *Multi-resolution localization of causal variants across the genome.* Cold Spring Harbor Laboratory. Retrieved from https://www.biorxiv.org/content/early/2019/05/24/631390 doi: 10.1101/631390

Sharma, A. (2017, May). *Understanding Activation Functions in Neural Networks.* Retrieved 02–09–2019, from https://medium.com/the-theory-of-everything/understanding-activation-functions-in-neural-networks-9491262884e0

Sitte, R., & Sitte, J. (2002, May 01). Neural networks approach to the random walk dilemma of financial time series. *Applied Intelligence*, *16*(3), 163–171. Retrieved from https://doi.org/10.1023/A:1014380315182 doi: 10.1023/A:1014380315182

Steyerberg, E., Vickers, A., Cook, N., Gerds, T., Gonen, M., Obuchowski, N., ... Kattan, M. (2010, 01). Assessing the Performance of Prediction Models: A Framework for Traditional and Novel Measures. *Epidemiology*, *21*(1), 128-138. doi: 10.1097/EDE.0b013e3181c30fb2

Sun, X. (1999). *The Lasso and Its Implementation for Neural Networks* (Unpublished doctoral dissertation). University of Toronto, Toronto, Ont., Canada, Canada. (AAINQ45795)

Sundermeyer, M., Alkhouli, T., Wuebker, J., & Ney, H. (2014, October). Translation modeling with bidirectional recurrent neural networks. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)* (pp. 14–25). Doha, Qatar: Association for Computational Linguistics. Retrieved from https://www.aclweb.org/anthology/D14-1003 doi: 10.3115/v1/D14-1003

Sutskever, I. (2013). *Training Recurrent Neural Networks* (Unpublished doctoral dissertation). University of Toronto.

Waibel, A. H., Hanazawa, T., Hinton, G. E., Shikano, K., & Lang, K. J. (1989). Phoneme recognition using time-delay neural networks. *IEEE Trans. Acoustics, Speech, and Signal Processing*, *37*, 328–339.

Werbos, P. J. (1990). Backpropagation through time: what it does and how to do it. *Proceedings of the IEEE*, *78*(10), 1550–1560.

Williams, R. J., & Zipser, D. (1989). A Learning Algorithm for Continually Running Fully Recurrent Neural Networks). *Neural Computation*, *1*, 270–280.

Yümlü, S., Gürgen, F. S., & Okay, N. (2005, October). A Comparison of Global, Recurrent and Smoothed-Piecewise Neural Models for Istanbul Stock Exchange (ISE) Prediction. *Pattern Recogn. Lett.*, *26*(13), 2093–2103. Retrieved from https://doi.org/10.1016/j.patrec.2005.03.026 doi: 10.1016/j.patrec.2005.03.026