Institute for Food and Resource Economics

University of Bonn

Discussion Paper 2023:1

A critical assessment of neural networks as meta-model of a farm optimization model

Claudia Seidel, Linmei Shang, Wolfgang Britz

Editor: Thomas Heckelei

Institute for Food and Resource Economics

University of Bonn

Nußallee 21

53115 Bonn, Germany

Phone: +49-228-732332

Fax: +49-228-734693

E-mail: thomas.heckelei@ilr.uni-bonn.de

Agricultural and Resource Economics, Discussion Paper 2023:1

A critical assessment of neural networks as meta-model of a farm optimization model

Claudia Seidel, Linmei Shang, Wolfgang Britz

Abstract

Mixed Integer programming (MIP) is frequently used in agricultural economics to solve farm-level optimization problems, but it can be computationally intensive especially when the number of binary or integer variables becomes large. In order to speed up simulations, for instance for large-scale sensitivity analysis or application to larger farm populations, meta-models can be derived from the original MIP and applied as an approximator instead. To test and assess this approach, we train Artificial Neural Networks (ANNs) as a meta-model of a farm-scale MIP model. This study compares different ANNs from various perspectives to assess to what extent they are able to replace the original MIP model. Results show that ANNs are promising for meta-modeling as they are computationally efficient and can handle non-linear relationships, corner solutions, and jumpy behavior of the underlying farm optimization model.

Keywords: artificial neural network, meta-model, mixed integer programming, farm optimization model

JEL classification: C45, C63, Q12

1 Introduction and problem background

Many decisions in real-life are optimization problems. For instance, farmers seek to maximize profits taking into account resource and policy constraints, potentially including leisure time considerations. Modern algorithms for Linear Programming (LP) allow to solve such farm-scale optimization problems quite fast subject to a large number of constraints and variables. They can depict in detail different farm branches and related machinery and building requirements. However, in order to accurately represent real-world farm management, indivisibilities in production and investment need to be taken into account, such as the construction of new stables, taking up off-farm work, full-time or part-time, or opting into specific policy programs. Simulation of such discrete choices require Mixed Integer Programming (MIP) (Britz et al., 2016), which is far more computing intensive and results often in a solution space containing more zero observations, jumps and indivisibilities compared to LP.

Observed (optimizing) real-world behavior of single agents is often characterized by discrete choices, which can be depicted by integer variables in MIP but not by differentiable functions. This is why MIP problems are used to depict decisions of farmers in Agent-based Models (ABMs) (Gilbert, 2007). Linking agent-specific MIP models into an ABM enables to simulate spatially explicit interactions of agents characterized by complex decision-making over distinct timesteps (Huber et al., 2018; Kremmydas et al. 2018). Applying this approach to spatially explicit land markets, such as in AgriPoliS (Happe et al., 2006), allows simulating emergent phenomena like structural change or technology adoption (Appel and Balmann, 2019; Huber et al., 2018; Shang et al., 2021; Shang et al. 2023).

The usual approach directly integrates a MIP model in the ABM (Britz, 2013; Happe et al., 2006; Schreinemachers and Berger, 2011; Troost and Berger, 2016). The MIP is then solved for each agent and time step (Troost and Berger, 2016; Britz, 2013). The optimal solution of a MIP problem depends on the interaction of its discrete variables. While finding a single MIP solution is typically relatively easy, proofing that such a solution is the best one is far harder. In extremum, it can require to assess all combinations of values that the set of discrete variables can take. This explains why MIP problems are NP-hard (nondeterministic polynomial time), meaning the solution time increases exponentially in model size. As a result, the direct integration of MIP into an ABM is highly computing time intensive (Troost and Berger, 2016; Schreinemachers and Berger, 2011; Happe et al., 2006). Consequently, ABMs can either only cover a small number of agents, or require reducing the complexity of the farm model compared to evolved stand-alone farm-scale models which can comprise several ten thousand equations and variables, of which several hundredths are binary or integer variables, as for instance in Britz et al. (2016).

Both options limit the explanatory power of the ABM. Meta-models (also called surrogate models) may be a way out of this problem. Meta-models are constructed to represent the simulation behavior of a complex simulation model through simpler mathematical functions and can be applied without optimization (Hussain et al. 2002; Reis dos Santos and Reis dos Santos, 2008). They can be used to verify, validate, optimize the parameterization or predict simulation models (Reis dos Santos and Reis dos Santos, 2008). The important characteristic of meta-models is their simplicity and speed (Kleijnen and Sargent, 2000).

In agricultural economics, large, often spatially explicit datasets, for instance depicting the land use history of individual plots, become increasingly available and allow for more evolved modeling approaches simulating agricultural structures and processes at larger scale. However, despite increasing computational power and improved algorithms, depicting decisions of each single farmer in a large farmer population in detail and linking them in a spatially explicit manner space remains challenging. Meta-modeling allows here to speed up ABMs to a point where they can be applied to large farming populations while maintaining the complexity of the underlying farm optimization model (Seidel and Britz, 2019; Storm et al. 2020; Troost et al. 2022). This makes it possible to depict both the highly detailed individual responses and interactions of many farmers, for instance, to simulate structural change dynamics in response to changes in technologies, markets or policies.

Duality-based econometric meta-models are a potentially inviting avenue. As an example, a dual profit function can be estimated. The resulting simulation equations are the first-order conditions of the underlying optimization problem and guarantee internally consistent responses in the quantity and value space. Duality estimation models have a long history of being applied to micro-observations, such that well-tested estimation approaches are available. Seidel and Britz (2019) explore this approach to develop a surrogate model of the bio-economic farm-scale model FarmDyn, developed at the University of Bonn, Germany (Britz et al., 2016). FarmDyn is MIP based and quite evolved. Specifically, the authors estimate a duality-based Symmetric Normalized Quadratic (SNQ) value function. The resulting estimated equations allow to simulate profits, related optimal input and output quantities as well as shadow prices of resources at given prices and resources. They form the meta-model which delivers the key results otherwise retrieved from solving a MIP in an ABM. While the authors achieved an overall satisfying fit for the key outputs, the meta-model failed to depict the jumpy behavior of the MIP in certain parts of the decision space. Such jumps cannot be captured by the smooth response of the relatively simple functional form of the SNQ value function. Furthermore, to be able to build a duality-based value function, they had to exclude corner solutions where input and output (in the following summarized as netput) quantities become zero or where resources are non-binding, for instance, when shadow prices of land are zero.

These two related aspects motivate the search for alternatives to duality-based approaches. In recent years, Artificial Neural Networks (ANNs) have been increasingly explored for meta-modeling as they are highly effective in approximating complex, non-linear relationships (Goodfellow et al., 2016; Gorr, 1994; Günther and Fritsch, 2010; Nezhad and Mahlooji, 2014; Razavi 2021; Storm et al. 2020). An ANN trained on a set of inputs and simulation results of a farm optimization model might overcome the restrictions when using simpler functional forms. Although studies exist in which ANNs are trained as meta-models of simulation models in agriculture or bioeconomy (e.g. Carnevale et al., 2012; Liong et al. 2001; Nguyen et al., 2019), the application of ANNs to imitate complex agricultural farm-level models in agricultural economics is a relatively new field of research (e.g. Shang et al., 2023; Troost et al. 2022). We want to extend this field of research by constructing and comparing different ANNs to represent the full response space of an evolved MIP model.

The objectives of this paper are a) to present the development of an ANN as a meta-model of an evolved MIP model, and b) to investigate how well ANNs can represent a MIP model with special regard to the solution behavior of MIP models. The structure of the paper is as follows. First, we present the general methodological approach of an ANN as a meta-model of a MIP model. In order to underline the necessity of an evolved meta-model, some details on the farm optimization model FarmDyn and its simulation behavior are presented. We then introduce the approach to train, tune and test different ANNs as meta-models of FarmDyn. In the fifth chapter, we present the results. We search for the best-fitted ANN first manually and then using a hyperparameter optimization algorithm. Afterwards, we investigate the suitability of an ANN as a meta-model of FarmDyn regarding its fit and run time, before we briefly conclude.

2 ANN as meta-model of MIP models

MIP based models do not only process but also produce potentially large amounts of data as outputs. The interpretation or further use of this data in other models can become an extensive task. In order to be able to run many simulations, for instance for large sensitivity analyses or, as in the discussed use case, in an ABM, a surrogate model can depict the most important input-output relationships of the MIP using mathematical functions. This has the advantage of requiring far less computational power. If the meta-model is able to imitate its underlying simulation model to a sufficient extent, the meta-model can replace the original model in simulations (Hussain et al. 2002; Reis dos Santos and Reis dos Santos, 2008).

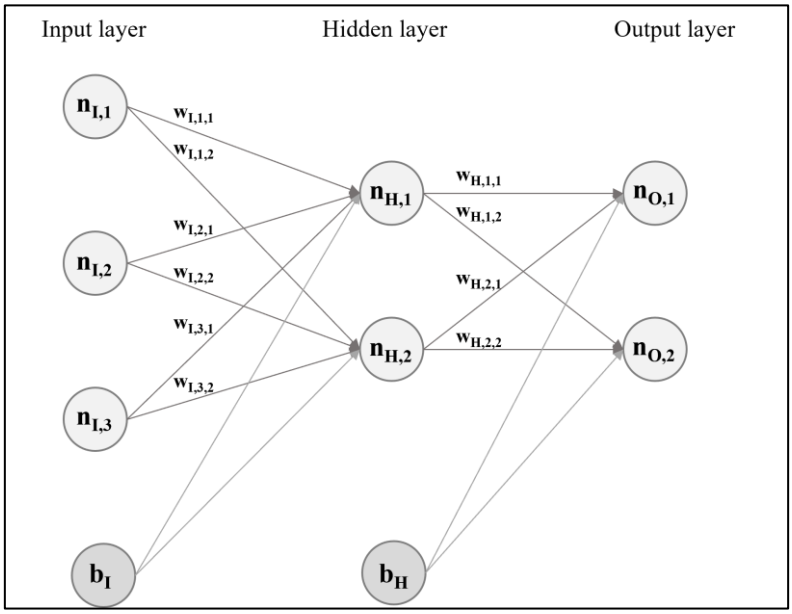
The idea of machine learning is that the algorithm itself learns from data a suitable model set-up to statistically estimate complicated non-linear functions (Goodfellow et al., 2016; Hastie et al., 2009). Within the large class of machine learning techniques, ANNs are very promising to act as meta-models of complex MIP models as they can handle non-linear structures and multiple outputs, and they are

much more computationally efficient than MIP models (Gorr, 1994; Storm et al. 2020). Given an appropriate learning algorithm, they can be highly accurate. Van der Hoog (2019) therefore suggests that an ANN can be trained to predict the behavior of agents in an ABM. He calls this the "micro-emulation" of agents' actions in ABMs by an ANN (van der Hoog, 2019, p. 1253).

An ANN is a machine learning algorithm, inspired by the function of a brain. The classical type of ANN is the multilayer perceptron which consists of several layers, each consisting of multiple units (neurons). The first layer of the neural network is the input layer, and the final layer is the output layer. The intermediate layers are called hidden layers because their output is not visible (Figure 1). An ANN with more than one hidden layer is also called a Deep Neural Network (DNN) because the magnitude of layers allows deep learning of the computer. The computer learns by building complex concepts expressed in terms of simple concepts represented by every single layer (Goodfellow et al., 2016).

An ANN works as follows. In a dense (or fully-connected) layer, each unit in the layer is linked to each unit in the previous one. In feedforward networks, which are widely applied ANN, information flows from neurons in the input layer, through intermediate computations in one or several hidden layers to the output layer. Weights and biases represent the parameters of an ANN. In each layer and for each neuron, the outputs of the previous layer are linearly combined based on their neuron specific weights. This weighted sum plus bias is then transformed by a non-linear activation function to define the neuron's output (figure 2) (Goodfellow et al., 2016; Günther and Fritsch, 2010; Hastie et al., 2009). The functions of the layers are connected, mostly in a chained structure, to create the final outputs (Goodfellow et al., 2016).

Figure 1: Structure of a simple fully-connected feedforward neural network

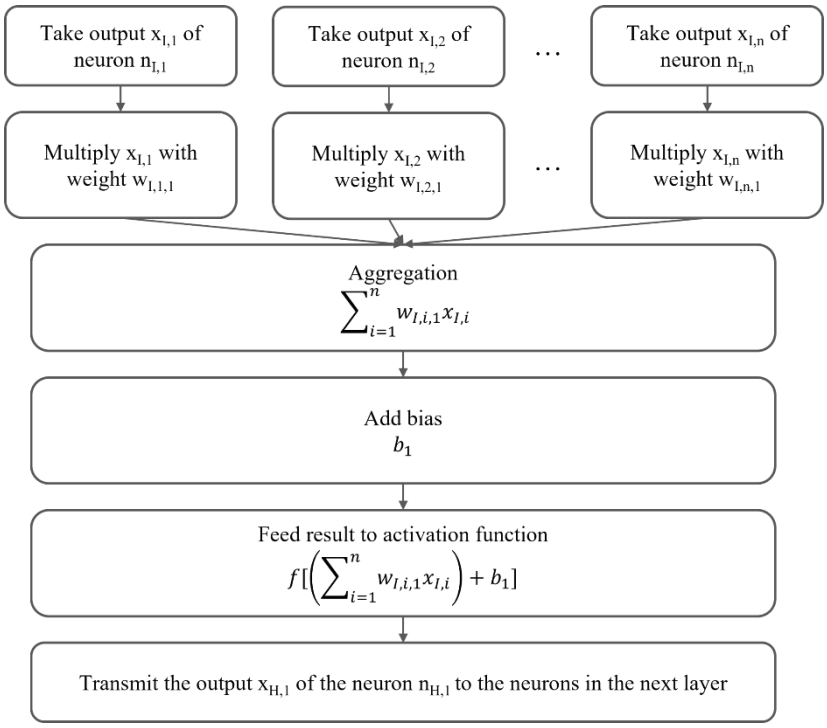


Source: own figure based on Goodfellow et al. (2016)

Note: $n_{I,i}$ are the i neurons of the input layer, $w_{I,i,j}$ are the weights of the i neurons of the input layer that are feed into the j neurons of the hidden layer $n_{H,j}$, $w_{H,j,k}$ are the weights of the i neurons of the hidden layer that are feed into the k neurons of the output layer $n_{O,k}$ and b_l are the biases that are feed into the neurons of the hidden layers and the output layers.

An ANN can hence be understood as a special case of a statistical estimation. It maps an input vector *x* to an output vector *y*, dependent on unknown parameters. However, in contrast to most econometric approaches, an ANN comprises typically far more estimated parameters as a weight is attached to each neuron, and an ANN imitating complex relationships may consist out of several hundreds of neurons. No particular assumptions, e.g. the distribution of error terms, need to be made. Furthermore, it is possible to include both qualitative and quantitative variables. As a consequence, ANNs allow for capturing multiple interactions of variables, non-linear relationships between variables and heterogeneous data (Pierreval, 1996; Storm et al., 2020).

According to the universal approximation theorem (Hornik et al., 1989), a feedforward network with a linear output layer and one hidden layer provided with enough hidden units and with any "squashing" activation function is able to approximate any measurable function from one finite-dimensional space to another with any desired non-zero error. However, these one-layer ANNs need potentially a large number of neurons for a satisfactory fit, i.e. a number exponential to the input dimension. Due to the large width of such networks, they may fail to learn. ANNs with more than one hidden layer are universal approximators as well, even with a limited number of neurons (depth-bounded). Thus, an ANN's depth is said to be more efficient than its width (Cybenko, 1989; Goodfellow et al., 2016; Hornik, 1991).Figure 2: Exemplary process flow of the first neuron in the input layer in a feedforward ANN



Source: own figure based on Goodfellow et al. (2016)

Considering these properties, we consider an ANN a promising choice to mimic the solution behavior of the MIP model. The exogenous drivers of the MIP model FarmDyn (resources, input and output prices, farm-specific technology parameters etc.) form the input features of the ANN and while its optimized key results, such as selected input and output quantities, profit and shadow prices, act as the target values. The ANN then aims to approximate the behavior of the MIP model by learning to map the input features, i.e. the exogenous parameters of a specific run of the MIP model, to the resulting optimized target values. Once the ANN is trained, it only needs a set of run specific inputs to predict

outputs by applying the learnt relationship. Therefore, the ANN can replace the MIP model in the ABM, as the inputs characterize a farm under specific market, policy and technology conditions.

3 The MIP model FarmDyn

FarmDyn (Britz et al., 2016) is a farm-scale bio-economic MIP model developed at the University of Bonn, Germany. It simulates economic optimal production and investment decisions in specific farm settings, typically the given resource endowment and considered activities, subject to given prices, technologies and policy instruments. The optimization targets farm-household income stemming for farm profits, including agricultural subsidies, and earnings from off-farm work, subject to a detailed representation of the production feasibility set, the maximum willingness to work on the farm or off-farm, liquidity restrictions and constraints given by policy regulations (Britz et al., 2016). Different farm branches and modules (dairy, mother cows, beef fattening, pig fattening, piglet production, arable farming, biogas plants) as well as policy restrictions (i.e. Greening, agri-environmental schemes, fertilization ordinance) can be added in a modular form to the optimization (Britz et al., 2016; Britz et al. 2021).

FarmDyn captures field operations, manure application, machinery etc. in high detail based on detailed engineering data for Germany which increases the plausibility of its simulation results for the different farm branches (Britz et al., 2021). The farm branches for dairy and cattle farming include different raising and fattening processes depending on the month, grazing share, weight gains, and calving and lactation period (Pahmeyer and Britz, 2020). Different farming activities and branches interact with each other. For instance, dairy and cattle farming options are interlinked with grassland management and arable farming including the consideration of different crop shares and rotations, production systems and intensity levels, manure disposal, storage capacity, biogas plants etc. (Kuhn et al., 2020; Schäfer et al., 2017). Investments in machinery, stables, silos, as well as off-farm work possibilities are depicted by integer variables (MIP model) to consider increasing returns-to-scale in branch sizes (Britz et al., 2021).

FarmDyn can be set up in a modular fashion. For our study, we choose the deterministic variant of FarmDyn which assumes a fully informed agent without consideration of risk. The model is run as a comparative static problem such that the model maximizes the farm household income of one year while herd dynamics are replaced by a steady-state model. Investment costs are annualized while the binary character of investments is maintained (Britz et al., 2016). Under these conditions, FarmDyn is solved for an existing sample of dairy farms used in Seidel and Britz, 2019.

The sampling of the model instance for FarmDyn is based on Latin Hypercube Sampling (LHS) (Britz et al., 2016; Iman and Conover, 1980; McKay et al., 1979). This efficient quasi-random sampling

procedure allows the whole range of factor level combinations to be represented in the sample. We assume a uniform distribution over each considered factor. Sampling is conducted in R, using the package of Carnell (2016). By varying the input values over a large range of different values, we obtain a dataset which represents the MIP's behavior on a larger set of different input values. The MIP model delivers optimal farm household income, netput quantities and marginal returns to land (shadow prices of land) (Britz et al., 2016). Marginal returns to arable and grassland (shadow prices of land) are not taken as reported by the solver as they are derived given the active set of integer variables. Instead, two follow up optimization runs are conducted where the land endowment for arable and grassland is exogenously increased by one ha. The resulting changes in profits compared to the given endowments define the marginal values (Seidel and Britz, 2019).

The industry MIP solver CPLEX 12.6 is used on a 44 core computing server under efficient solution strategies, such as parallel computing on multiple cores and reducing the solution space by removing non-active decision variables under a relaxed MIP (RMIP) solution. A detailed description of the structure and solution procedure of FarmDyn is documented in the model description (Britz et al., 2016).

4 Building an ANN as a meta-model

This section details the development of an ANN as a meta-model of a MIP model, here FarmDyn. The meta-modeling process consists of three basic steps. First, farm experiments are simulated using FarmDyn; second, the ANN is trained to imitate FarmDyn and its parameters are tuned to optimally fit it to our FarmDyn settings; third, ANN is used to simulate surrogate results for FarmDyn using a test dataset.

4.1 *Generating MIP experiments*

The first step in the meta-modeling approach generates a sufficiently large set of outcomes from the MIP model FarmDyn. We embed the study in an agricultural intensive area in North-Rhine Westphalia (NRW) in Germany where dairy farming plays a dominant role. These farms are highly specialized and can be assumed to be rationally managed. The profitability of dairy farming is mainly determined by feeding costs and milk prices, but also rental prices of farmland play a crucial role in a potential farm size expansion of dairy farms (Bradfield et al. 2023; Frick and Sauer, 2021; Margarian, 2010; Zimmermann and Heckelei, 2012). That is why we use the dairy and arable farming modules of FarmDyn and choose levels for prices and endowments that correspond to intensive dairy farms.

Based on the price and structural statistics of NRW (IT.NRW, 2022; KTBL, 2016), factor ranges of explanatory variables such as farm sizes, number of animals etc. are defined. A dairy farming

population is defined with Design of Experiments (DOE). For each observation of this farming population, a MIP model is solved (Britz et al., 2016).

A large dataset is needed for the development of an ANN as a meta-model of FarmDyn. By varying the input values over a large range of different values, a dataset is obtained which represents the MIP model's behavior on a large variety of different input sets. Specifically, we let FarmDyn solve three million times.

We delete observations with negative values of shadow prices of arable land and shadow prices of grassland. This can happen if all farm labor is used in the optimal solution and an additional hectare is forced to be managed, as also idling land requires some inputs. Furthermore, we exclude the 1% and 99% quantile of profits, shadow prices of arable land and shadow prices of grassland in order to exclude unrealistic observations from analyses (since FarmDyn is used to represent data close to real-world data), while maintaining a large variety of different farm endowments. The reduced dataset contains 1049710 observations. Our study will contain an analysis of how the dataset size influences the capability of our ANN to represent FarmDyn in our setting and investigate whether one million observations are sufficient to represent a complex MIP model.

A challenge that all meta-models face is the choice of variables extracted from the original simulation model (Pierreval, 1996). The ANN must be provided with enough inputs and outputs of the MIP model to be able to learn the behavior of the MIPs. In order to catch all information necessary to represent FarmDyn by a meta-model, we select the most important in- and outputs of the FarmDyn dataset, based on literature (Bradfield et al. 2023; Frick and Sauer, 2021; Margarian, 2010; Zimmermann and Heckelei, 2012) and own sensitivity analyses of FarmDyn. As inputs, we choose the prices of milk, manure export, cattle feed, feeding crops, off-farm labor, and investments, as well as the (quasi-)fixed factors cows per farm working unit, farm working units, arable land, grassland and milk yield. As outputs, we select the amount of milk produced, manure exported, cattle feed and feed crops bought, off-farm labor supplied, and investments, as well as the profit, shadow price of arable land and shadow price of grassland. In total, we have eleven inputs and nine outputs (Table 1).

Prior to training, the dataset needs to be normalized in order to stabilize and speed up the training process (Goodfellow et al., 2016). We apply min-max normalization in Python.

The dataset is divided into training and test data. The training dataset is used for the identification and training of the ANN while the test dataset is used for ANN prediction and serves us for the investigation of the performance of the ANN as meta-model of FarmDyn in Chapter 6. The test dataset consists of 20% of the observations or 209942 observations, which leaves a training dataset of 839768 observations.

Table 1: Inputs and outputs of FarmDyn used for ANN meta-modeling

Inputs		*Outputs*
Netput prices	*(Quasi-)fixed factors, technology*	
Manure export price	Cows per farm working unit	Profit
Cattle feed price	Farm working units	Marginal return to arable land
Price for feeding crops	Hectares of arable land	Marginal return to grassland
Investment price	Hectares of grassland	Quantity of manure exported
Milk price	Milk yield	Quantity of cattle feed bought
Wage of off-farm work		Quantity of feeding crops bought
		Quantity of investments made
		Quantity of raw milk produced
		Amount of off-farm work

Sources: own table

4.2 *Building and training of the ANN*

Next, we build the model and train it to approximate the behavior of the MIP model. Similar to other statistical models, the aim is to minimize the difference between the output predictions of the ANN and the given outputs from the MIP.

The number of neurons in the input and output layer is equal to the number of exogenous inputs and output variables chosen from the FarmDyn dataset. Consequently, we have eleven inputs, resp. neurons in the input layer and nine outputs, resp. neurons in the output layer. The number of neurons in the hidden layers as well as the number of layers itself is not pre-defined and part of the ANN tuning procedure as presented in the subchapter 4.2.1. We build the ANN in Python using the neural network library Keras (Gulli and Pal, 2017).

We train the ANN in a supervised way. Based on the comparison of the FarmDyn output y with the ANN output $\hat{y}$ given in the training dataset, a cost function (or loss function, error function) is calculated. This cost function is iteratively minimized by adjusting the weights of the neurons. During training the ANN learns the relationship between inputs and outputs by trying to match the predicted output of the ANN to the output given in the training dataset (Goodfellow et al., 2016; Hastie et al., 2009). In analogy to the minimum least squares optimization in linear regression, a common way is to compute the cost as the Mean Squared Error (MSE) of the normalized y,

$$MSE = \frac{1}{N}\sum_{i}^{N}\sum_{j}^{C}(\hat{y}_{ij} - y_{ij})_i^2$$

(1),

where N is the number of observations, C is the number of outputs of the model, $\hat{y}_{ij}$ is the predicted outcome returned by the model and y_{ij} is the observed outcome for the ith training case and the jth output (Goodfellow et al., 2016).

Feedforward ANNs are trained using back-propagation. The back-propagation algorithm calculates first partial derivatives of the cost function with respect to each weight in the ANN. The first partial derivatives indicate how quickly the error changes if weights and biases change. The minimization of the cost function is done by gradient descent. The partial derivatives calculated by the back-propagation algorithm represent the slope of the cost function. The idea of gradient-descent is to move in the direction of the steepest descent of the cost function. In iterative steps, weights are adjusted in a way to move down the cost function in the direction with the steepest descent (Goodfellow et al., 2016; Hastie et al., 2009).

Depending on the particular research question, other fit measures exit or adjustments of the MSE measure are possible. For example, the mean absolute error can be used to give lower weights to extreme outliers. Another possibility is to introduce output specific weights if one particular output is especially important (Goodfellow et al., 2016; Hastie et al., 2009).

4.2.1 ANN tuning

According to the literature, a simple one-layer one-neuron ANN is able to learn complex behavior. Adding layers to the ANN improves the in-depth learning of the ANN and leads to efficiency gains (Goodfellow et al., 2016). Besides the number of hidden layers, the performance of an ANN depends on other hyperparameters. These are inter alia the number of neurons in the hidden layer, the so-called dropout rate, weight initialization, optimizer, activation function, learning rate, momentum, number of epochs, minibatch size, and weight decay coefficient (Goodfellow et al., 2016). Their theoretical influence on the performance of an ANN is presented in Table 2.

Hyperparameters are not adapted by the learning algorithm during training but are pre-determined by the modeler in order to adapt the algorithm to the specific problem (Goodfellow et al., 2016). Failures in the application of ANNs can be avoided by reasonable choices of the hyperparameters (Hornik et al., 1989). Hyperparameters need to be chosen to render the ANN precise enough to capture the complex

relationship between inputs and outputs, but at the same time general enough to be able to generalize to other input datasets, i.e. avoid overfitting (Goodfellow et al., 2016).

Table 2: Hyperparameters tested and their typical influence on the performance of ANN

Hyperparameters	*Typical impact on ANN performance*
Number of hidden layers	Number of layers increases the capacity of ANN in capturing complex underlying relationships between inputs and outputs, thus increasing the accuracy of an ANN.
Number of hidden units (neurons)	Similar to the number of hidden layers, the number of hidden units also increases accuracy. A small number of units per layer causes underfitting.
Dropout rate	Dropout is a technique to reduce overfitting. At each training iteration, a fraction of input units is randomly set to 0 implying that some information learned is randomly removed. Small dropout rates show often limited effects; while high values can result in under-learning. Usually, a value between 20% and 50% is chosen.
Weight initialization	Weights are initialized before training starts and optimized from there. Typically, weights are randomly initialized. If weights are initialized with very high or very low values, training might be not efficient (vanishing/exploding gradient problem). The choice interacts with the chosen activation function. Some weight initialization techniques have been developed (He optimization, Xavier optimization). Some activation functions like ReLU and leaky ReLU solve the problem of vanishing gradient.
Optimizer	The optimization algorithm finds the weights and biases that minimize the loss function during training. Stochastic Gradient Descent (SGD) and its variants are frequently used optimizers for deep learning.
Activation function	The activation function is used to introduce non-linearity in the estimation. The rectified linear activation function (ReLU) and sigmoid function are recommended default functions for feedforward ANNs.
Learning rate	The learning rate determines the size of the step along the cost curve during gradient descent training. A low learning rate lets the training converge smoothly, but drives up the required time. A high learning rate might speed up the process, but convergence might fail.
Momentum	The momentum term introduces a dependency between weight update and last weight update, thereby preventing oscillations in the learning process and preventing getting stuck in a local minimum. Momentum is usually between 0.5 and 0.9.
Number of epochs	One epoch means that the whole dataset is passed forward and backward through the ANN once, i.e. the number of epochs is the number of times the entire training dataset runs through the network during training. A low number of epochs leads to underfitting;

Hyperparameters	*Typical impact on ANN performance*
	a high number of epochs to overfitting. As a rule of thumb, the number of epochs should be increased until the validation accuracy starts decreasing.
Minibatch size	Dividing the sample into subsamples and training ANN iteratively on each of the batch makes the model more resistant to outliers and variance on the training set. The typical default batch size is 32. Typically, batch sizes are chosen in exponential steps, i.e. 32, 64, 128, 256 and so forth.
Weight decay coefficient	An additional term in the weight update rule that causes the weights to exponentially decay to zero, if no other update is scheduled. It is a regularization mechanism used to avoid overfitting.

Sources: based on Goodfellow et al. (2016)

To find the best combination of hyperparameters is an extensive task for modelers. To speed-up the research, hyperparameter tuning procedures have been developed. They test pre-defined ranges of different hyperparameter levels and look for the best combinations based on a specific error rate (Bergstra and Bengio, 2012; Probst et al. 2019). As hyperparameter optimization algorithms are highly computationally intensive, manual tuning is often the first step to limit the possible combinations of hyperparameter settings in a follow-up automatized hyperparameter optimization procedure.

In order to investigate the influence of various hyperparameters on our ANN, we first test different hyperparameter settings step-by-step. In a large sensitivity test framework, we vary ceteris paribus (c.p.) the number of neurons, minibatch size, learning rate, choice of the optimizer, activation function and the number of epochs. Furthermore, we c.p. add layers to the ANN to improve the in-depth learning of the ANN. Thus, we transform the ANN into a DNN. The influence of the dataset size on the accuracy of the ANN is also investigated.

Hyperparameters are selected using a validation dataset which we generate by randomly splitting 10% of the training dataset. The model will not be trained on his data. The validation dataset is used to test the model performance during training by calculating the MSE and R^2. Based on these two metrices, the best combination of hyperparameters tested is identified. The basic setting of hyperparameters used in the c.p. sensitivity analyses as well as the various tested settings are presented in Table 3.

Table 3: Model specification for sensitivity analysis

Hyperparameters	*Basic setting*	*Tested settings*
Number of hidden layers	1	1-10
Number of neurons	64	8-1024 in exponential steps
Minibatch size	32	8-512 in exponential steps
Learning rate	0.001	0.0001, 0.0005, 0.001, 0.005, 0.01
Optimizer	Adam	Adam, Adagrad, Adamax, SGD, RMSprop
Activation function	ReLU	Sigmoid, tanh, LeakyReLU, ReLU
Number of epochs	100	50, 100, 150, 200

Sources: own table

4.3 *ANN prediction*

The test dataset can be used to make predictions in order to investigate the ANN performance. If the ANN is not able to make accurate predictions for the test dataset, even though the fit of the train dataset was high, the model is overfitted. If the predictions of the test dataset are as accurate as of the test dataset, the ANN is ready for use.

The trained ANN can be applied in the ABM for each agent, depicted by its endowment, technology and its market environment according to the eleven inputs, to predict production and investment decisions and profits in each time step. While training the ANN can be computationally intensive, once the network is trained, it allows very fast prediction (van der Hoog, 2019). The short inference time speeds up the computation of the agents' decision-making in the ABM compared to solving a farm model in each time step. If external prices or endowments change, only the inputs to the ANN change. The learned relationship between inputs and outputs remains the same.

Farmers' behavior in the ABM, including production decisions, investment decisions, technology adoption, expanding or quitting agricultural production stems from the ANN. This behavior reflects profit maximization at given prices, farm endowments and policy regulations, and generates corresponding shadow prices of land.

5 Results

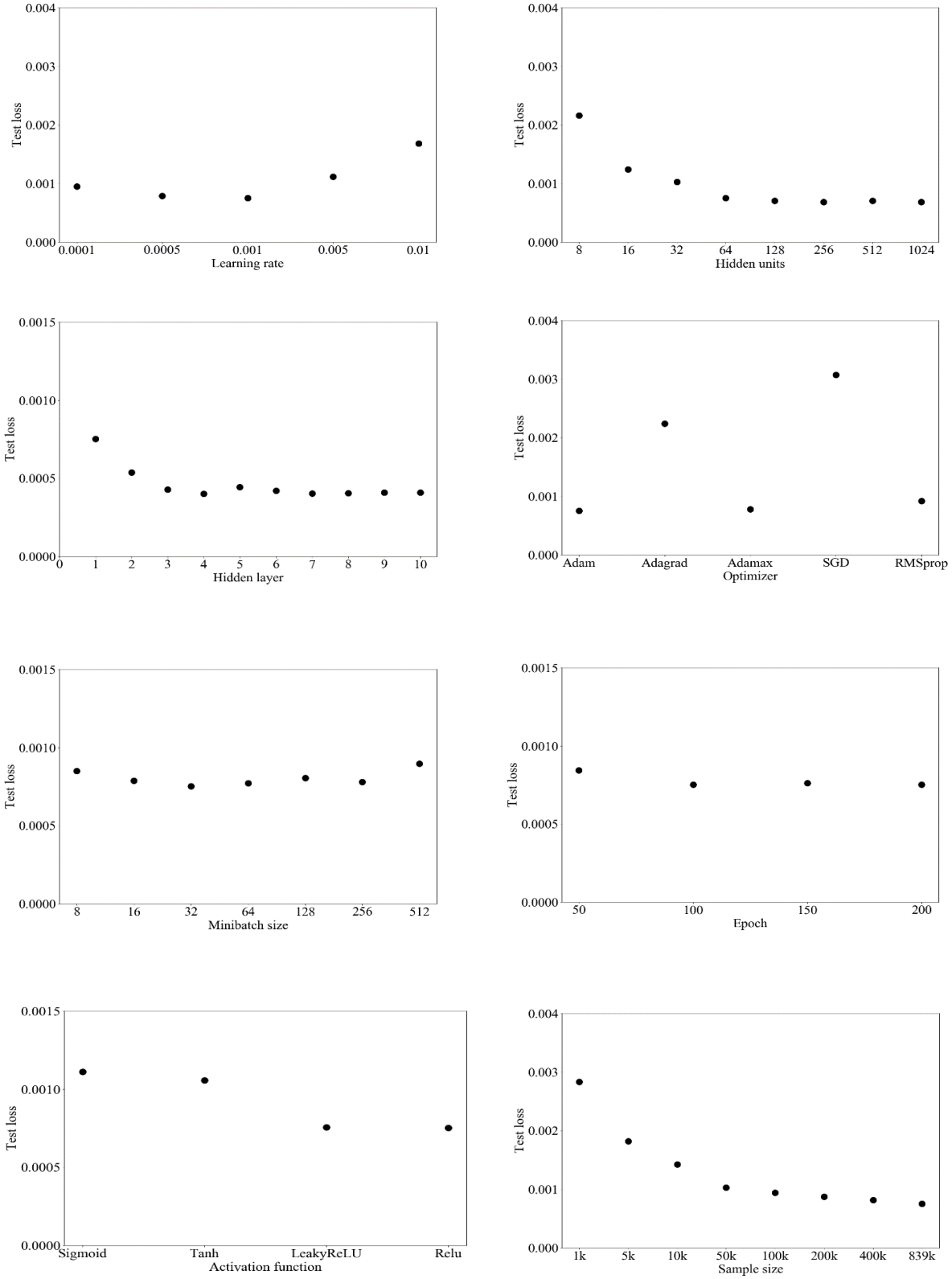
We look for the optimal set of hyperparameters for our application in two ways: first, manually by running a sensitivity analysis of the model with various model settings (section 5.1) and second, applying a hyperparameter optimization algorithm (section 5.2) using the optimal hyperparameter settings found in the manual tuning.

5.1 *Results of manual tuning*

Manual tuning allows a precise analysis of the effects of different hyperparameter settings. We can investigate whether the theoretical impacts of hyperparameters on the performance of ANN can also be detected in our application.

Figure 3 presents the results of the sensitivity analyses displaying the MSE of the test dataset for each hyperparameter tested. In the following, we present the results considering the MSE and R^2 of the test dataset as indicators of model fit.

Figure 3: Test loss of different hyperparameter settings in manual tuning



Sources: own results

- Learning rate

  In our experiments, the running time did not change depending on the learning rate. The accuracy of ANN, however, changes. We get the lowest MSE and highest R^2 on our test dataset with a learning rate of 0.001.

- Units in the hidden layer

  In order to test the universal approximation theorem for our ANN, we increase the number of neurons starting from eight neurons in exponential steps up to 1024 neurons in the hidden layer. The fit of the ANN increases as the number of neurons goes up. The best result is as expected achieved at the considered maximum of 1024 hidden units. As can be seen in figure 3, the performance gain is relatively strong when the number of neurons increases from 8 to 64. From there, the accuracy increases only very slightly. With a total R^2 of 0.9770, the fit of the ANN with 64 neurons to the MIP model is very high. With only ten neurons, R^2 of the test set is 0.8534. With a higher number of neurons, it slightly increases up to 0.9795. Especially, the variable off-farm work which is an integer variable in FarmDyn cannot accurately be depicted by an ANN with ten neurons. The R^2 lies at 0.2694. With 64 and more neurons, the ANN is able to accurately represent the integer character of the variable ($R^2 > 0.99$).

- Hidden layers

  We increase the number of layers iteratively from one up to ten. The R^2 of the test set is lowest for a one-layer ANN but still on a very high level (0.9770 for one layer compared to 0.9868 for four layers). The ANN with four layers achieves the lowest MSE. More complex ANNs perform slightly worse or almost equally. This might reflect other limits in the training process. An explanation might be the number of epochs chosen for manual tuning. 100 epochs might be too small to allow accurate training for an ANN with five and more hidden layers. This consideration displays the interlinkage of hyperparameters and its influence on training performance.

- Optimizer

  All tested optimizers are extensions of SGD. The choice of the optimizer has a strong influence on the performance of the ANN (Goodfellow et al., 2016). Adam, Adamax and RMSprop achieve significantly better results than SGD and Adagrad. With a test loss of 0.000752 and R^2 of 0.977, Adam is the best choice for our application.

- Minibatch size

  In our experiments, a batch size of 32 leads to the lowest MSE which is the typical default batch size of ANN. The running time is 23 seconds per epoch. R^2 is not affected by the batch size. In all settings, we attain a total test R^2 of around 0.97-0.98.

- Epochs

  The smallest tested number of epochs achieves the best result. 100 epochs seem to be the best choice in our experiment. Test total R^2 is 0.977. With another number of epochs, this does not change significantly.

- Activation function

  Out of the four tested activation functions (sigmoid, tanh, leakyReLU and ReLU), ReLU leads to the best result. The test total R^2 of leakyReLU and ReLU is almost at the same level, i.e. 0.9773 and 0.9770. The best candidate, ReLU, is defined by the function $g(z) = max\{0, z\}$, where $z = W^T x + b$ (Goodfellow et al., 2016).

- Sample size

  Figure 3 also shows the influence of the training dataset size on the performance of the ANN. The ANN was trained with dataset sizes of 1000, 5000, 10000, 50000, 100000, 200000, 400000 and 839768. As expected, the loss MSE decreases with increasing dataset size from 0.0028 to 0.0007. However, the fit improves very slightly for sizes between 100000 and 839768 observations. If the data generating process is time and resource intensive, it may hence be sufficient to use a smaller training dataset from a MIP model. We gather that at least 100000 observations are necessary to represent the dairy farming branch of FarmDyn by an ANN. Adding additional branches will probably ask for a larger dataset.

  R^2 improves from 0.8980 in case of 1000 observations to 0.9770 in case of 839768 observations. The improvements especially relate to the shadow prices of land, export quantity of manure and amount of off-farm labor. For instance, the R^2 of the shadow price of arable land increases from 0.7478 with 1000 observations to 0.9119 with 839768 observations.

To sum up, our findings confirm the influence of hyperparameters found in the literature. Since we tested changes in one hyperparameter at given default choices for all others, we might miss the best combination of hyperparameters. The results from manual tuning are delegated to an automatized tuning algorithm in the next chapter.

5.2 *Results of the hyperparameter optimization algorithm*

The manual hyperparameter tuning shows the influence of each single hyperparameter on the goodness of fit, at unchanged values of all others. To advance here, we apply Hyperopt (Bergstra, et al. 2013), a Bayesian optimization algorithm in Python which searches for an optimal choice of hyperparameters for a machine learning model by an automatic tuning procedure.

In order to speed up Hyperopt, we restrict the optimization to the number of neurons in each layer and use otherwise the best hyperparameters from the manual tuning. We take a learning rate of 0.001, sample size of 839768, mini-batch size of 32, Adam as optimizer, and ReLU as activation function in each hidden layer and in the output layer. Due to computational constraints, Hyperopt searches for optimal network structures with up to five hidden layers. Table 4 shows the structures of the optimal ANN (i.e. the number of neurons in each hidden layer) and their performances depending on the number of hidden layers.

Table 4: Structures and performances of different ANNs found by Hyperopt

	1 hidden layer		*2 hidden layers*		*3 hidden layers*		*4 hidden layers*		*5 hidden layers*	
Optimal number of neurons in each hidden layer	H1:	1750	H1: H2:	602 383	H1: H2: H3:	1011 940 1431	H1: H2: H3: H4:	1111 658 1249 711	H1: H2: H3: H4: H5:	1724 829 1804 754 1334
Train loss	0.00068		0.00051		0.00044		0.00039		0.00038	
Test loss	0.00068		0.00052		0.00044		0.00040		0.00038	
Train R^2	0.9794		0.9835		0.9854		0.9789		0.9791	
Test R^2	0.9794		0.9835		0.9854		0.9791		0.9792	
Time needed to predict 1 million data points	15.64 s		98.81 s		131.73 s		95.32 s		291.67 s	

Sources: own results

The number of neurons in each layer could range from 8 to 2048. Table 4 shows that the model with three hidden layers and 1011 neurons in the first, 940 neurons in the second and 1431 neurons in the third hidden layer has the highest R^2 on the training and test data set. The train and test loss metrics are quite low, but somewhat below the ones of ANNs with four or five hidden layers. The time needed to predict one million data points is considerably higher for the ANN with three layers compared to ANNs with one, two or four hidden layers. The latter comes as a surprise as the total number of neurons in the version with four layer exceeds the one with three.

6 Investigation of the ANN as meta-model of FarmDyn

The in Table 4 presented values of MSE and R^2 indicate that the ANN with three hidden layers fits best to the FarmDyn settings used in the study. Table 5 summarizes the optimal hyperparameter settings as identified by manual and automatized tuning.

Table 5: Best ANN found by manual tuning and hyperparameter optimization

Hyperparameters	*Best ANN*
Number of hidden layes	3
Number of neurons	1011, 94 1431
Minibatch size	32
Learning rate	0.001
Optimizer	Adam
Activation function	ReLU
Number of epochs	100

Source: own results

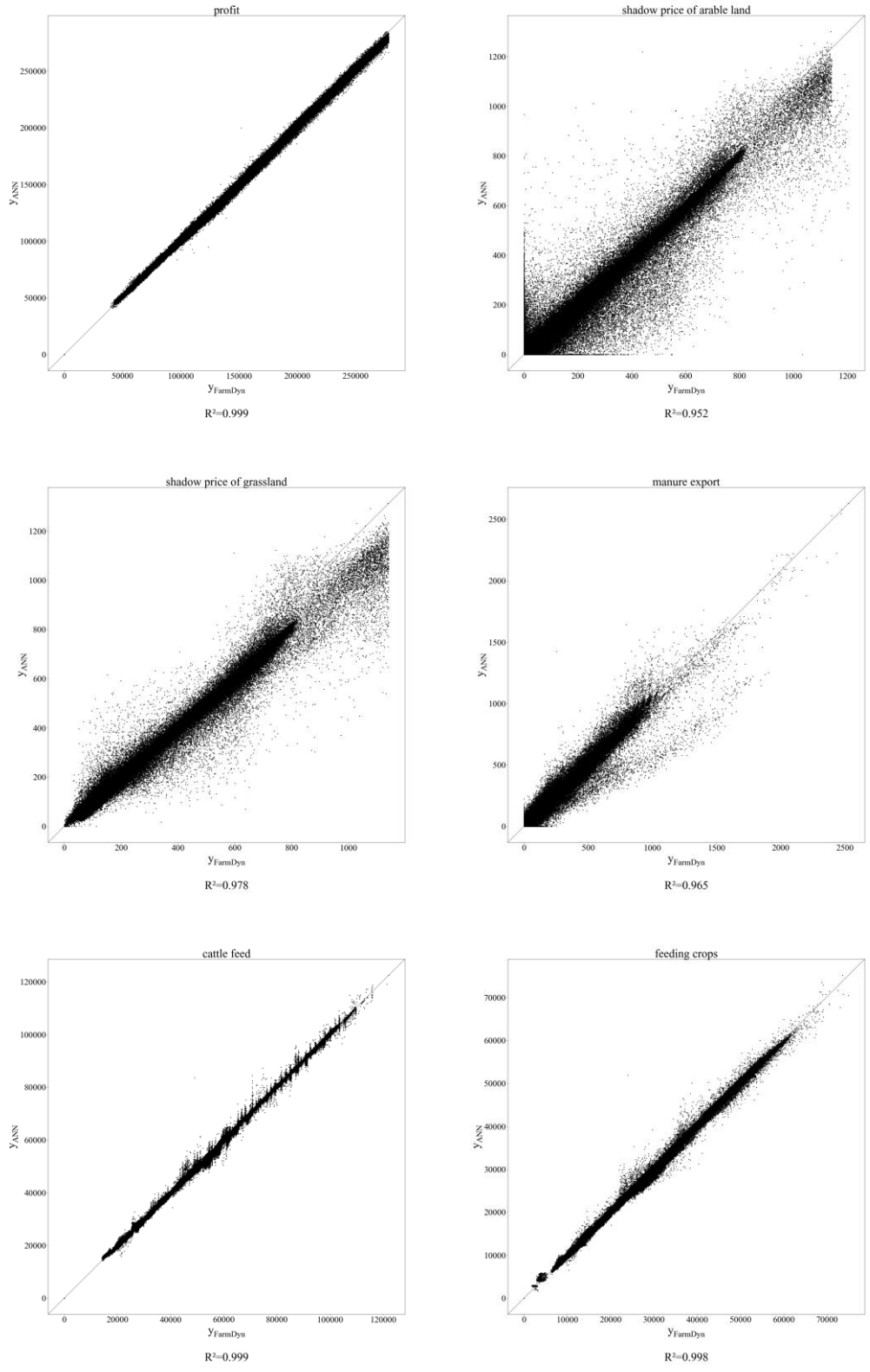
While a high fit is required to make an ANN a promising candidate to replace the MIP in model linkage, Shang et al. (2023) propose three other metrics to consider: first, whether fundamental relationships between an input and an output or between two outputs what they term consistency of bivariate relationships, second, accuracy in capturing corner solutions and third, accuracy in holding constraints, such as ensuring that simulated crop acreages exhaust total land. Moreover, the authors stress that ANNs are suitable candidates for meta-modeling only if not an excessive number of observations is needed to train them.

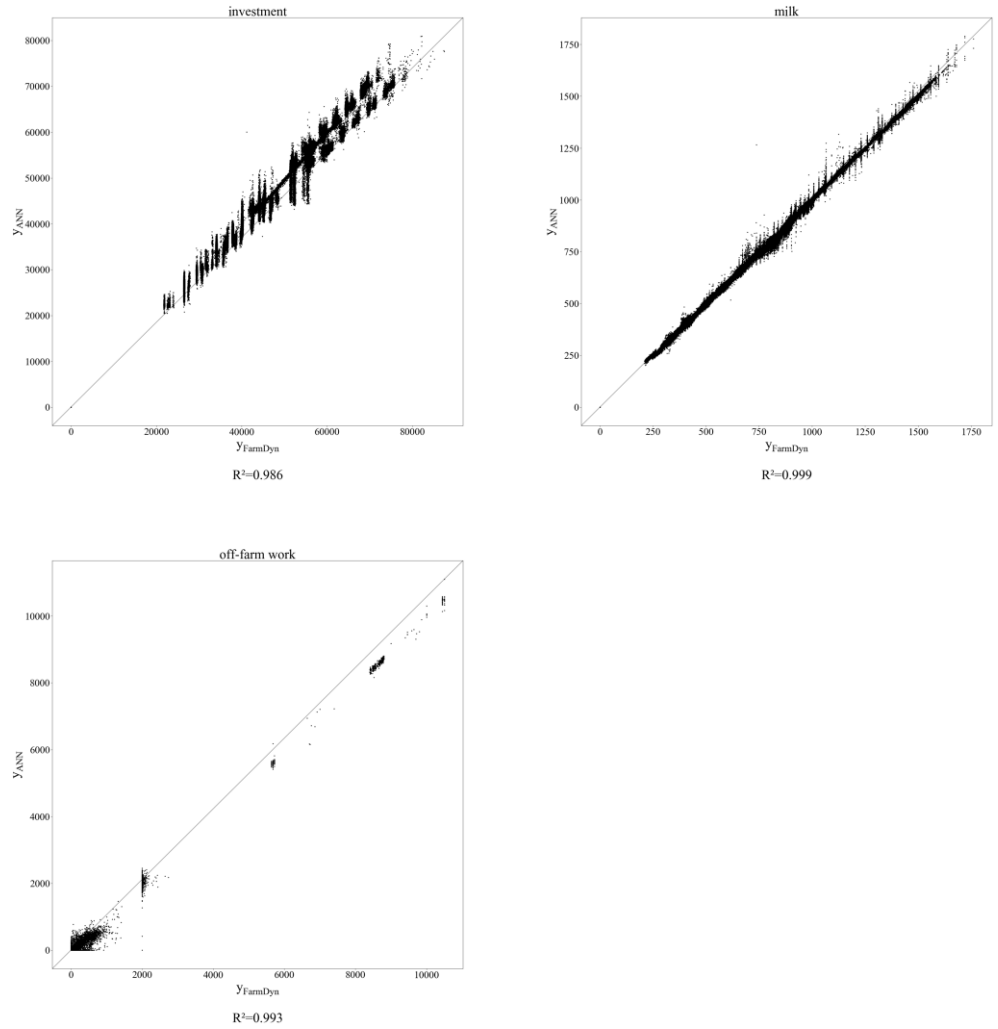
We investigate the applicability of our ANN based on the goodness of fit, the capability of replicating individual farm decisions and running time considerations.

6.1 *Considering the model fit*

As suggested by the high average MSE, the scatterplots in figure 4 show that the ANN is able to represent outputs of a complex MIP model with a high degree of accuracy, with a R^2 between 0.953 and 0.999 for the outputs using the test material consisting of 209942 observations.

Figure 4: Scatterplots of predicted outputs





Source: own results

A specific challenge in meta-modeling of MIPs is an appropriate representation of outputs (strongly) linked to integer variables. As we can see from the scatterplots, this is the case especially for investments and off-farm work where the spikes indicating integer solutions are well recognizable. Similar patterns can be observed for milk output and the use of cattle feed. This reflects that the quantity of milk produced, and linked to this the amount of feed needed, are determined by the herd size at given milk yield. The herd size in turn depends on the chosen stable size to a large degree, as farmers tend to fully use the available stable places in order to maximize profits. In the version of FarmDyn used for this study, stables are available in predetermined sizes, provoking jump discontinuities. Despite their non-differentiable nature, the capacity of the ANN to represent integers and related netputs is high with an R^2 between 0.993 and 0.999.

If an agricultural ABM integrates a MIP, the bidding behavior of farmers in land markets is typically informed by simulated land shadow prices. The achieved R^2 of 0.952 for the marginal of arable land and of 0.978 for grassland are quite high, but lower than for the considered netputs.

6.2 *Representing decision-making of an individual farm*

The R^2 indicate that the overall fit of the ANN is clearly satisfactory, but there might still be cases where predictions by the ANN and the true simulated results might considerably deviate. We therefore test the ANN in sensitivity analysis for a single farm, under changing costs of manure exports and changing endowments with grass land (see Table 6, also for value of other inputs). The results of ANN prediction are compared to the output of FarmDyn in figure 5 and figure 6.

Table 6: Experiment settings

	Experiment grassland	*Experiment manure export price*
Manure export price (€/m³)	5.0	1.0 – 13.0
Cattle feed price (€/t)	280.0	280.0
Price for investment	1.0	1.0
Milk price (cent/liter)	33.5	33.5
Off-farm wage (€/hour)	8.0	8.0
Price for feeding crops (€/t)	200.0	200.0
Annual working units	2.6	2.6
Ha of arable land (ha)	39.0	39.0
Ha of grassland (ha)	30.0 – 35.0	30.0
Number of cows	100.0	100.0
Milk yield (100 kg/head)	84.0	84.0

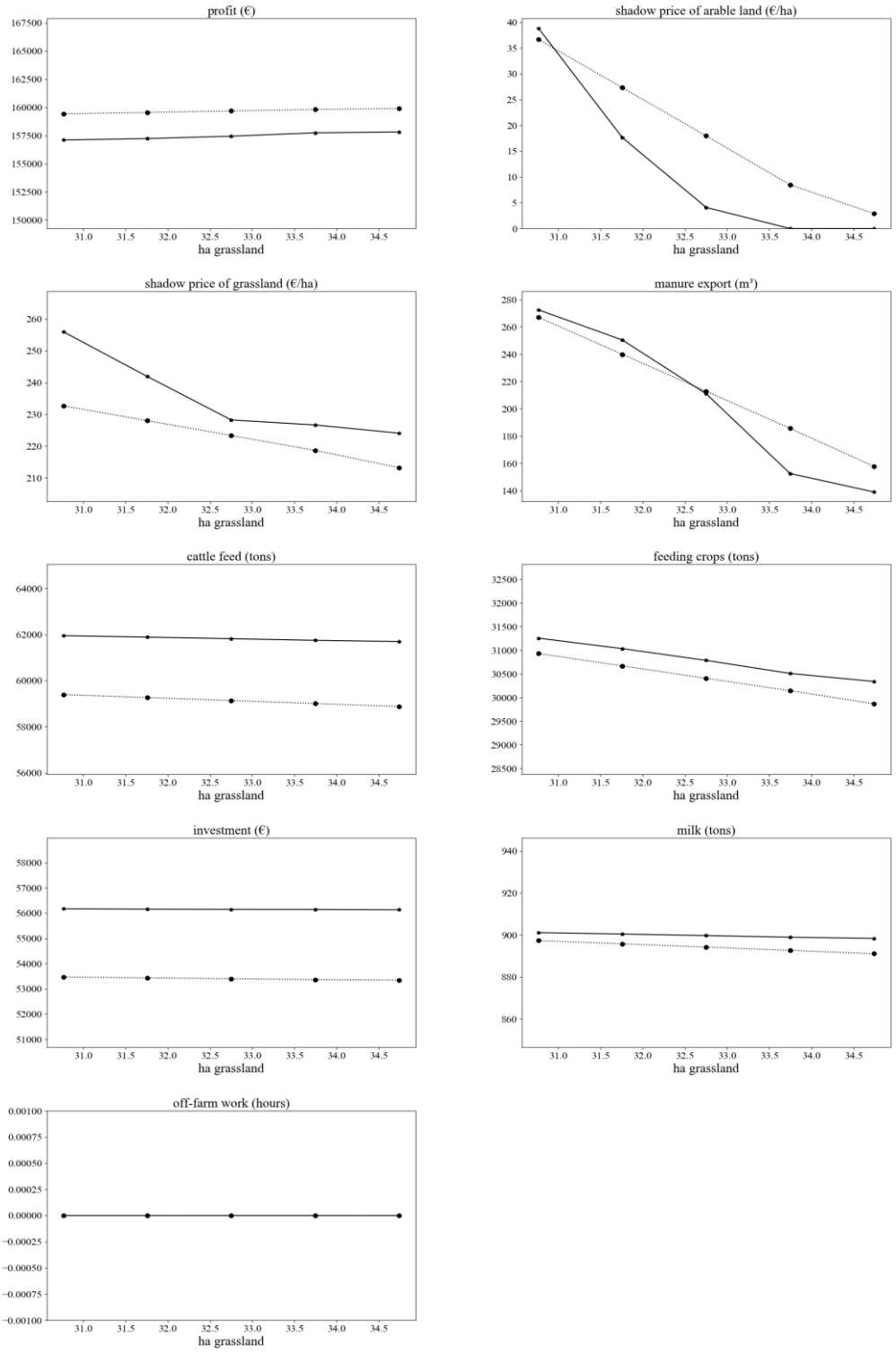
Note: Prices of cattle feed, resp. feeding crops are price averages of various feeding concentrates, resp. crops.

The panes in figures 5 and 6 underline that the ANN predicts the FarmDyn netputs in both experiments quite accurately and also generates response curves of the dependent variables quite close to the MIP (profit, shadow prices of arable land, shadow price of grassland, as well as quantities of manure export, cattle feed, crops for feeding, milk, investments, and off-farm work). In both experiments, the farmer does not work off-farm which is also predicted by the ANN.

In both experiments, the ANN tends to underestimate somewhat netput quantities and to overestimate profits. Results for shadow prices differ: in the manure export experiments, both land shadow prices are underestimated, while with increasing grassland endowments, shadow prices of arable land are overestimated and of grassland underestimated. Still, the curves of shadow prices of arable land

as well as the quantity of manure export show that the ANN is able to predict jumps in the solution space of FarmDyn. This also holds for corner solutions. First, the ANN correctly predicts that in both experiments the farmer does not work off-farm. Second, in the manure export price experiment, the ANN is able to predict the jump from 0 to a shadow price of land < 100 with a change in the manure export price from 1 to 3 € per m^3. Overall, these tests suggest that the ANN is a promising candidate to replace the MIP as a meta-model.

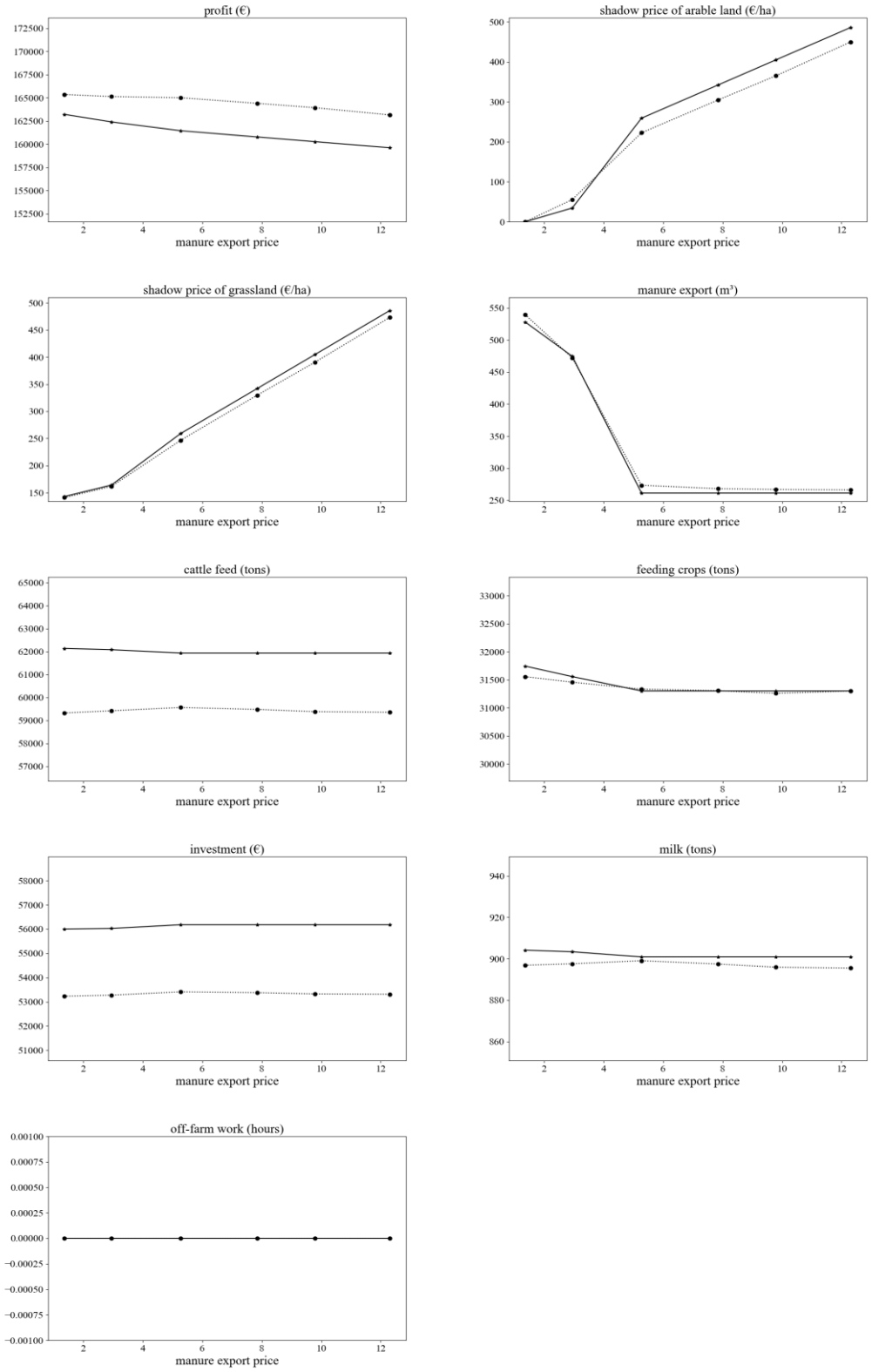
Figure 5: Results of grassland experiment



Note: ── FarmDyn output, ····· ANN output

Sources: own results

Figure 6: Results of manure export price experiment



Note: ── FarmDyn output, ····· ANN output

Sources: own results

6.3 *Considering run time*

ANNs are characterized by very fast predictions (van der Hoog, 2019). As shown above, the selected three-layer ANN is able to predict one million scenarios in slightly more than two minutes. In comparison, the FarmDyn required several weeks to generate a data set with 1 million observations. Run time differences between the different ANN tested in there are considerable. A one-layer ANN with 1750 neurons requires solely 16 seconds or around 15% for the same predictions, and still achieves a quite high fit (measured by R^2 and MSE on the test set). Depending on the targeted application of the ANN, a slight loss in accuracy might be accepted given this speed gain.

ANNs require larger datasets for training and testing, and the one-time effort to produce the observations must be offset by gains in later application. Moreover, which fit can be achieved with a given training set size is not known beforehand. Equally, the training sets must be carefully constructed to cover the later input range. Designing the construction of the training sets and related computational burden might discourage modelers from using an ANN as a meta-model (Nguyen et al., 2019).

For our study, as found during manual tuning, we gather that at least 100000 observations are needed to train an ANN with a satisfactory fit for our MIP model. While the fit still increases up to the maximum of more than 800000 observations considered in our tests, the improvements are quite muted. This implies that one should step-wise add new observations to the training set and repeatedly train the ANN to find a compromise between computational needs to generate observations and the achieved fit. In this process where only observations are added, the additional training time is far lower compared to a start from scratch. The same holds when the input range must be expanded at some later point, as it was in our case for herd sizes beyond 1000 cows.

Not only the step to generate training data, but also the training process itself is time demanding. Training time can be decreased by adjusting hyperparameters such as optimizer, number of layers and neurons, and learning rate (Goodfellow et al., 2016).

7 Conclusion

In this study, we constructed ANNs as meta-models of the evolved MIP model FarmDyn, which is used to simulate production and investment decisions of intensively managed dairy farms. The main advantage of using the meta-model instead of the MIP model in simulations is the tremendous gain in speed. This opens up the door to extend the regional or temporal scale in modeling approaches. Moreover, software aspects might favor the relatively simple simulation approach based on the trained equations of the ANN over integrating a MIP model in the overall modeling framework.

This study shows that meta-modeling based on ANNs is superior in terms of approximating accuracy, replicating corner solutions, and capturing jumps in the decision space, compared to standard

econometric meta-modeling approaches in case of MIP models. We reached a R^2 of more than 0.95 for profits, shadow prices of land and netput quantities, despite a quite jumpy behavior of the original MIP for many of these outputs. Test simulations with arbitrarily chosen starting endowments and netput prices showed that the ANN correctly depicts corner solutions and jumps. The latter will be usually not the case with duality based econometric approaches which are restricted by the nature of their functional forms. We therefore conclude that an ANN based meta-model could be used in ABMs to depict a large set of heterogeneous farms, replacing the evolved MIP from which is derived. The results of our study thus confirm existing literature. If it is pre-defined and trained in an appropriate way, an ANN can learn to mimic complex models. We achieved the best results with a three-layer DNN by manual tuning, this set-up was confirmed by subsequent hyperparameter optimization.

The main disadvantage of using ANNs in meta-modeling of a MIP is the large amount of observations and the time needed for training. Generating training data can require weeks of computing time, even if modern industry MIP solvers and efficient solution techniques are applied, and requires a careful design of the observation space to properly capture the later ranges of inputs in simulations. These steps and the tuning of ANN demand for additional know-how by the modeler. Finding the right balance between over- and underfitting of the ANN and the choice of hyperparameters is to a limited degree possibly based on textbook knowledge and using hyperparameter tuning algorithms, but requires expertise.

Once an ANN is trained to imitate the MIP model, its application in ABMs is very efficient as it makes fast and accurate predictions. This approach therefore has a high potential for agricultural research to be able to run large-scale and long-term simulations of agricultural policies and markets.

References

Appel, F. and A. Balmann (2019): Human behaviour versus optimising agents and the resilience of farms –Insights from agent-based participatory experiments with FarmAgriPoliS. *Ecological Complexity*, 40(B): 100731.

Bergstra, J. and Y. Bengio (2012): Random search for hyperparameter optimization. *Journal of machine learning research*, 13: 281-305.

Bergstra, J., Yamins, D., and D.D. Cox (2013): Making a Science of Model Search: Hyperparameter Optimization in Hundreds of Dimensions for Vision Architectures. TProc. of the 30th International Conference on Machine Learning (ICML 2013), June 2013, pp. I-115 to I-23.

Bradfield, T., Butler, R., Dillon, E.J., Hennessy, T., and J. Loughrey (2023): The impact of long-term land leases on farm investment: Evidence from the Irish dairy sector. *Land Use Policy*, 126: 106553.

Britz, W. (2013): ABMSim – A flexible framework for Agent Based Models to simulate spatially explicit structural change in agriculture, Methodological and technical documentation, Version 1.0. University of Bonn, Germany, http://www.ilr.uni-bonn.de/em/rsrch/abmsim/abmsim_version1.pdf, (accessed March 07, 2023).

Britz, W., Ciaian, P., Gocht, A., Kanellopoulos, A., Kremmydas, D., Müller, M., Petsakos, A., and P. Reidsma (2021): A design for a generic and modular bio-economic farm model. *Agricultural Systems*, 191: 103133.

Britz, W., Lengers, B., Kuhn, T., and D. Schäfer (2016): A highly detailed template model for dynamic optimization of farms-FARMDYN University of Bonn Institute for Food and Resource Economics, Version Sept, 147, 2016.

Carnell, R. (2016): lhs – Latin Hypercube Samples, R-Package, Version 0.13. https://cran.r-project.org/web/packages/lhs/index.html, (accessed August 22, 2022).

Carnevale, C., Finzi, G., Guariso, G., Pisoni, E., and M. Volta (2012): Surrogate models to compute optimal air quality planning policies at a regional scale. *Environmental Modelling and Software*, 34: 44–50.

Cybenko, G. (1989): Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals and Systems*, 2: 303-14.

Frick, F. and Sauer, J. (2021): Technological Change in Dairy Farming with Increased Price Volatility. *Journal of Agricultural Econonmics*, 72: 564-588.

Gilbert, N. (2007): Agent-Based Models (second edition). SAGE Publications Inc, London.

Goodfellow, I., Bengio, Y., and A. Courville (2016): Deep Learning, MIT Press, Cambridge, Massachusetts.

Gorr, W.L. (1994): Editorial: Research prospective on neural network forecasting. *International Journal of Forecasting*, 10(1): 1–4.

Gulli, A. and S. Pal (2017): Deep learning with Keras. Packt Publishing Ltd.

Günther, F. and S. Fritsch (2010): Neuralnet: training of neural networks. The R J., 2: 30 – 38.

Happe, K., Kellermann, K., and A. Balmann (2006): Agent-based Analysis of Agricultural Policies: an Illustration of the Agricultural Policy Simulator AgriPoliS, its Adaptation and Behavior. *Ecology and Society*, 11(1).

Hastie, T., Tibshirani, R., and J. Friedman (2009): The Elements of Statistical Learning. Data Mining, Inference, and Prediction, Second Edition, Springer Series in Statistics, Springer; New York.

Henningsen, A. (2020) micEconSNQP: Symmetric Normalized Quadratic Profit Function, R-Package, Version 0.6-6, https://cran.r-project.org/web/packages/micEconSNQP/micEconSNQP.pdf, (accessed August 22, 2022).

Hornik, K. (1991): Approximation capabilities of multilayer feedforward networks. *Neural networks*, 4(2): 251-257.

Hornik, K., Stinchcombe, M., and H. White (1989): Multilayer feedforward networks are universal approximators. *Neural Networks*, 2: 359–366.

Huber, R., Bakker, M., Balmann, A., Berger, T., Bithell, M., Brown, C., Grêt-Regamey, A., Xiong, H., Le, Q.B., Mack, G., Meyfroidt, P., Millington, J., Müller, B., Polhill, J. G., Sun, Z., Seidl, R., Troost, C., and R. Finger (2018): Representation of decision-making in European agricultural agent-based models. *Agricultural Systems*, 167: 143–160.

Hussain, M.F., Barton, R.R, and S.B. Joshi (2002): Metamodeling: Radial basis functions, versus polynomials. *European Journal of Operational Research*, 138: 142–154.

Iman, R.L. and W.J. Conover (1980): Small sample sensitivity analysis techniques for computer models, with an application to risk assessment (with discussion). *Communication in Statistics*, 9: 1749–1842.

IT.NRW (2022): Database of the statistical office of North-Rhine Westphalia, Germany, https://www.landesdatenbank.nrw.de/ldbnrw/online?operation=statistic&levelindex=0&levelid=1645612832604&code=41141#abreadcrumb, (accessed August 22, 2022).

Kleijnen, J.P.C. and R.G. Sargent (2000): A methodology for fitting and validating meta-models in simulation. *European Journal of Operational Research*, 120(1): 14–29.

Kohli, U. (1993): A Symmetric Normalized Quadratic GNP Function and the U.S. Demand for Imports and Supply of Exports. *International Economic Review*, 34(1): 243–255.

Kuhn, T., Enders, A., Gaiser, T., Schäfer, D., Srivastava, A.K., and W. Britz (2020): Coupling crop and bio-economic farm modelling to evaluate the revised fertilization regulations in Germany. *Agricultural Systems*, 177: 102687.

Kremmydas, D., Athanasiadis, I.N., and S. Rozakis (2018): A review of Agent Based Modeling for agricultural policy evaluation. *Agricultural Systems*, 164: 95–106.

KTBL (Association for Technology and Structures in Agriculture) (2016): Betriebsplanung Landwirtschaft 2016/2017, KTBL Datensammlung, Darmstadt, Germany.

Liong, S.-Y., Khu, S.-T., and W.-T. Chan (2001): Derivation of Pareto Front with Genetic Algorithm and Neural Network. *Journal of Hydrologic Engineering*, 6: 52–61.

Margarian, A. (2010). Coordination and differentiation of strategies: the impact on farm groth of strategic interaction on the rental market for land. *German Journal of Agricultural Economics*, 59(670-2016-45890): 202-216.

McKay, M.D., Beckman, R.J., and W.J. Conover (1979): A Comparison of Three Methods for Selecting Values of Input Variables in the Analysis of Output from a Computer Code. *Technometrics*, 21: 239.

Mohammad Nezhad, A., and H. Mahlooji (2014): An artificial neural network meta-model for constrained simulation optimization. *Journal of the Operational Research Society*, 65(8): 1232-1244.

Nguyen, T.H., Nong, D., and K. Paustian (2019): Surrogate-based multi-objective optimization of management options for agricultural landscapes using artificial neural networks. *Ecological Modelling*, 400: 1-13.

Pahmeyer, C. and W. Britz (2020): Economic opportunities of using crossbreeding and sexing in Holstein dairy herds. *Journal of Dairy Science*, 103(9): 8218 – 8230.

Probst, P., Boulesteix, A.L., B. and Bischl (2019): Tunability: Importance of hyperparameters of machine learning algorithms. *The Journal of Machine Learning Research*, 20(1): 1934-1965.

Razavi, S. (2021): Deep Learning, Explained: Fundamentals, Explainability, and Bridgeability to Process-based Modelling. *Environmental Modelling & Software*, 144: 105159.

Reis dos Santos, M.I. and P.M. Reis dos Santos (2008): Sequential experimental designs for nonlinear regression metamodels in simulation. *Simulation Modelling Practice and Theory* 16 (9): 1365-1378.

Schäfer, D., Britz, W., and T. Kuhn (2017): Flexible Load of Existing Biogas Plants: A Viable Option to Reduce Environmental Externalities and to Provide Demand-driven Electricity? *German Journal of Agricultural Economics*, 66(2): 109–123.

Schreinemachers, P. and T. Berger (2011): An agent-based simulation model of human-environment interactions in agricultural systems. *Environmental Modeling and Software*, 26(7): 845–859.

Seidel, C. and W. Britz (2019): Estimating a Dual Value Function as a Meta-Model of a Detailed Dynamic Mathematical Programming Model. *Bio-based and Applied Economics*, 8: 75–99.

Shang, L., Heckelei, T., Gerullis, M.K., Börner, J., and S. Rasch (2021): Adoption and diffusion of digital farming technologies - integrating farm-level evidence and system interaction. *Agricultural Systems*, 190: 103074.

Shang, L., Wang, J., Schäfer, D., Heckelei, T., Gall, J., Appel, F., and H. Storm (2023): Surrogate modelling of a detailed farm-level model using deep learning. *Journal of Agricultural Economics*, 00: 1– 26.

Storm, H., Baylis, K., and T. Heckelei (2020): Machine learning in agricultural and applied economics' *European Review of Agricultural Economics*, 47: 849–892.

Troost, C. and T. Berger (2016): Simulating structural change in agriculture: Modelling farming households and farm succession. International Environmental Modelling and Software Society (iEMSs), 8th International Congress on Environmental Modelling and Software, Toulouse, France.

Troost, C., Parussis-Krech, J., Mejaíl, M. & Berger, T. (2022): Boosting the scalability of farm-level models: efficient surrogate modeling of compositional simulation output. *Computational Economics*

van der Hoog, S. (2019): Surrogate Modelling in (and of) Agent-Based Models: A Prospectus. *Computational Economics*, 53: 1245–1263.

Zimmermann, A. and T. Heckelei (2012): Structural change of European dairy farms – A cross-regional analysis. *Journal of Agricultural Economics*, 63(3), 576-603.