Received: 2 May 2022 | Revised: 3 April 2023 | Accepted: 15 April 2023

DOI: 10.1111/1477-9552.12543

JAE Journal of Agricultural Economics

ORIGINAL ARTICLE

Surrogate modelling of a detailed farm-level model using deep learning

Linmei Shang[1] | **Jifeng Wang**[2] | **David Schäfer**[1] | **Thomas Heckelei**[1] | **Juergen Gall**[2,3] | **Franziska Appel**[4] | **Hugo Storm**[1]

[1]Institute for Food and Resource Economics (ILR), University of Bonn, Bonn, Germany

[2]Department of Information Systems and Artificial Intelligence, University of Bonn, Bonn, Germany

[3]Lamarr Institute for Machine Learning and Artificial Intelligence, Sankt Augustin, Germany

[4]Leibniz Institute of Agricultural Development in Transition Economies (IAMO), Halle (Saale), Germany

Correspondence
Linmei Shang, Institute for Food and Resource Economics (ILR), University of Bonn, Bonn Germany.
Email: linmei.shang@ilr.uni-bonn.de

Funding information
Deutsche Forschungsgemeinschaft, Grant/Award Number: EXC-2070-390732324-PhenoRob; European Commission, Grant/Award Number: 817566-MINDSTEP

Abstract

Technological change co-determines agri-environmental performance and farm structural transformation. Meaningful impact assessment of related policies can be derived from farm-level models that are rich in technology details and environmental indicators, integrated with agent-based models capturing dynamic farm interaction. However, such integration faces considerable challenges affecting model development, debugging and computational demands in application. Surrogate modelling using deep learning techniques can facilitate such integration for simulations with broad regional coverage. We develop surrogates of the farm model FarmDyn using different architectures of neural networks. Our specifically designed evaluation metrics allow practitioners to assess trade-offs among model fit, inference time and data requirements. All tested neural networks achieve a high fit but differ substantially in inference time. The Multilayer Perceptron shows almost top performance in all criteria but saves strongly on inference time compared to a Bi-directional Long Short Term Memory.

KEYWORDS

agent-based model, deep learning, farm modelling, neural networks, surrogate model, upscaling

JEL CLASSIFICATION

C45, C63, Q12, Q18

1 | INTRODUCTION

Modelling the impacts of agri-environmental policies increasingly requires accounting for detailed farm-level decision-making, heterogeneous local conditions, and interaction among farmers. Policies that are relatively homogeneous across regions (such as tariffs and export subsidies at the EU level or decoupled income support) are continuously substituted or complemented with more targeted farm-level policies—for example, the newly introduced eco-schemes or collective agri-environmental payments that require coordination and participation of local communities (Kuhfuss et al., 2016; Šumrada et al., 2022). Detailed farm-level models (Richardson et al., 2014; Weersink et al., 2002), usually implemented as optimisation models, are capable of representing individual decision-making with a rich representation of input choices, investments and environmental indicators. However, those farm-level models usually do not account for interaction among farmers, market feedback, or environmental feedback on larger scales (Heckelei, 2013; Shang et al., 2021). Here, agent-based models (ABMs) (Gilbert, 2007) can be used to model endogenous market feedback and to capture the dynamic interaction of heterogeneous farms (Kremmydas et al., 2018; Müller et al., 2020; Rasch et al., 2017). However, computational demands limit the complexity of farm decision-making models within an ABM or the number of agents and hence the regional coverage of those models (Bradhurst et al., 2016; Murray-Rust et al., 2014; Sun et al., 2016). Integrating detailed farm-level models as individual decision-making models into ABMs—while still covering a larger region—is desirable for policy analysis but usually causes high computational costs in application, difficulties in data exchange, and challenges in model update/debugging. We address this issue by training and evaluating computationally efficient surrogates that can be integrated into ABMs in place of the original farm models without any relevant losses in accuracy and detail of model outcomes.

We demonstrate the training and evaluation of surrogate models of the farm-level model FarmDyn (Britz et al., 2016), which could be integrated into ABMs. To make the discussion more concrete, we consider the ABM Agricultural Policy Simulator (AgriPoliS) (Appel & Balmann, 2019; Happe et al., 2006) as an example, but surrogate models could equally be used in other ABMs. FarmDyn is an economic simulation tool that is used ex-ante to assess agricultural policy reforms and the adoption of new technologies. It simulates farm production and investment decisions under changes in prices of inputs/outputs, technology and policy instruments for different farming branches in Germany and other countries (Britz et al., 2021). The linkage of biophysical parameters to highly detailed farming activities enables users to assess both economic and environmental policies with a wide range of social, economic and environmental indicators at the farm level. It has been applied, for example, to assess the impact of the revised German fertilisation ordinance (Kuhn et al., 2020), the impact of changes in water levels of peat soils on farm income (Poppe et al., 2021), the impact of European fertiliser laws on legume production (Heinrichs et al., 2021) and the potential adoption of a new pesticide additive (Kuhn et al., 2022). The profit-maximising solution of a farm is solved by Mixed-Integer Programming (MIP), which is time-consuming when many variables and constraints of different types are involved (Seidel & Britz, 2019).

AgriPoliS is a spatial and dynamic ABM that explicitly models farmers' interaction on the land market. It has been used to study the impact on agricultural structural change of different policies, such as decoupling direct payment (Happe et al., 2008) and Germany's biogas policy (Appel et al., 2016). In AgriPoliS, farmer agents maximise household income/profit, which is also solved by MIP. Compared to FarmDyn, the MIP in AgriPoliS is simpler because it models less detailed technology choices and faces fewer constraints (e.g., environmental constraints). Direct integration of the MIP in FarmDyn and AgriPoliS to combine the strengths of both is computationally demanding and quickly becomes prohibitive as the spatial coverage expands (Bradhurst et al., 2016; Huber et al., 2022; Sun et al., 2016). Besides, the two models running

together render model updates/debugging very challenging. However, combining the advantages of both types of models becomes increasingly necessary for agri-environmental policy analysis (Huber et al., 2018).

Surrogate models, also known as metamodels or emulators, may solve this problem (Jiang et al., 2020; Ratto et al., 2012). They approximate computationally costly simulation models by mapping the relationship between inputs and outputs while being much cheaper to run. The availability of highly flexible machine learning tools such as neural networks (NNs) (Goodfellow et al., 2016) offers the opportunity to build surrogates of complex and computationally demanding simulation models (Razavi, 2021; Storm et al., 2020). In this way, a surrogate model functions as a bridge between detailed farm-level models and large-scale ABMs to efficiently utilise the advantages of both types of models. Although parallel simulation on a high-performance computer (see An et al., 2021) can also save computational time, surrogate models provide several advantages: (1) High-performance computing is not always available or access might be limited, which can be a bottleneck, particularly during development; (2) Having a computationally more efficient surrogate model allows a larger number of experiments to be performed in a short time with the same computational resources; (3) Surrogate models allow a more natural separation of the development of the farm-level model and the ABM. This allows us to modularise the integrated modelling system, which can in the long run simplify model update/debugging and foster model reusability for the benefit of other researchers (Britz et al., 2021). It can also simplify collaboration between different research groups in terms of software licensing and data access issues. For example, the farm-level model might be run in a proprietary software environment that might not be available for the group running the ABM, while the surrogate model is trained in Python with all parts being open-access.

Surrogate modelling has been applied in various fields, such as water resource modelling (Razavi et al., 2012), engineering (Jiang et al., 2020), weather forecasting (Chen et al., 2020), and agricultural economics (Troost et al., 2022). Troost et al. (2022) develop different types of surrogate models to approximate a farm-level model using multinomial-logistic regression, multivariate adaptive regression splines, random forest regression and extreme gradient boosting. Their surrogate models capture the underlying relationship between 22 inputs (prices and model uncertainty parameters) and 9 outputs (crop areas). However, to our knowledge, the application of surrogate modelling using NNs in agricultural economics does not yet exist. In a broader sense, there are only two studies—Audsley et al. (2008) and Nguyen et al. (2019)—that use NNs to approximate a crop model and biogeochemical model to predict crop yields and soil organic carbon, which are further used in economic models. However, both studies use a classical type of NN, multilayer perceptron (MLP). To the best of our knowledge, surrogate models of a detailed farm optimisation model with different architectures of NNs is unexplored in agricultural economics. We develop surrogate models for FarmDyn as a first step such that it can be integrated into ABMs like AgriPoliS.

We see four main contributions. First, we show it is possible to build well-fitted surrogates of detailed farm-level models using NNs. Second, we systematically compare the performances of different architectures of NNs. Third, we develop a set of evaluation metrics to assess the quality of surrogate models. Here, we go beyond criteria such as R^2 or mean squared error (MSE) and develop generic metrics that can also be applied to evaluate other surrogate models. They help judge if the trained surrogate provides the required accuracy for the intended purposes. This is essential because different NN architectures deviate substantially in inference time (i.e., the time to make one prediction) with only minor differences in R^2 or MSE. Thus, more detailed and practically relevant evaluation metrics are required to judge if those differences in R^2 or MSE are of practical importance and justify the increased inference time. Fourth, we investigate the performance of surrogate models given different amounts of training data to provide practical guidance for modellers. While it is possible to increase the amount of data by running the underlying model deliberately, it is often computationally expensive. Hence, for

practical purposes, it is crucial to determine how much data is required for different architectures of NNs to achieve the desired performance on the defined evaluation metrics.

This rest of the paper is organised as follows. Section 2 reviews existing surrogate models to identify the common architectures of NNs currently used in the literature. Section 3 introduces the overall research design. In Section 4, we analyse the results and assess the performance of NNs given different amounts of training data. Section 5 provides a conceptual discussion on using surrogate models in ABMs for agricultural policy simulation. The last section concludes and points out directions for future research.

2 | NNS AS SURROGATE MODELS IN THE LITERATURE

Surrogate models in the literature are based on a large variety of model types, including polynomial regression (Hussain et al., 2002), radial basis functions (Amouzgar & Strömberg, 2017), kriging (Kleijnen, 2009), Gaussian processes (Picheny, 2015), support vector machines (Xiang et al., 2017), genetic programming (Fallah-Mehdipour et al., 2013), Bayesian networks (Gruber et al., 2013) and NNs (Sun & Wang, 2019). Throughout this paper, we focus on NNs as they bring new promise for surrogate models that require lower computational cost (Chen et al., 2021). This section introduces basic concepts of NNs and identifies the common architectures of NNs used as surrogates in the literature. Note that the approach of replacing agents' decision-making with NN-based surrogate models is different from the approach that uses NNs as underlying structure in ABM (e.g., Jäger, 2021).

2.1 | Basic concepts of NNs

NNs are capable of representing highly non-linear relationships and are well placed to deal with high dimensions in the input and the output space. Figure 1a depicts the most commonly used architecture of NN: MLP. It consists of an input layer, an output layer, and at least one hidden layer between the two. Each layer contains a certain number of neurons. Like a biological neuron, an artificial neuron processes the information from the inputs in the previous layer and transfers the signal to the next neuron, as shown in Figure 1b. An artificial neuron performs two steps of computation. First, a weighted sum of all inputs is computed as shown in Equation (1):

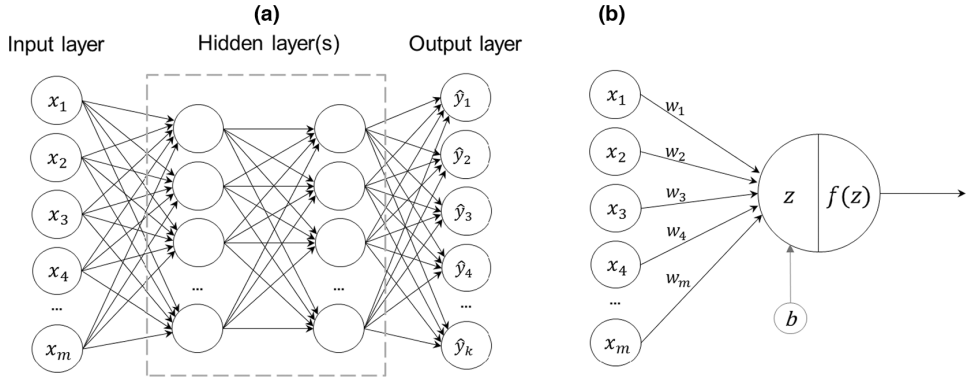

FIGURE 1 The architecture of an MLP (a) and an artificial neuron (b). x_i is the value of an input neuron, $\hat{y}_i$ is the prediction of an output neuron, w_i is the weight of a neuron, b is the bias, z is the output of the weighted sum, and $f(z)$ represents the activation function. *Source*: Based on Goodfellow et al. (2016).

$$z = \sum_{i=1}^{m} w_i x_i + b \quad (1)$$

where w_i is the weight of the input neuron x_i, b is the bias, m is the number of input neurons, and z is the weighted sum.

Second, the weighted sum will be transferred by an activation function ($f(z)$). Typically, activation functions are non-linear. For example, the Rectified Linear Unit (ReLU) returns the value that is equal to the input if it is positive, and it returns zero otherwise.[1]

Weights and biases are called 'parameters' of an NN. Training an NN like MLP finds the optimal parameters to minimise the loss function, that is, a function that measures the difference between the predicted outputs and the simulated outputs (e.g., the MSE loss). This process is usually done iteratively through backpropagation algorithms that compute the gradient of the loss function with respect to the weights and biases (Rumelhart et al., 1986). The gradients are then used by an optimisation algorithm (an optimiser) to update the parameters. Training the NN with all the training data for one cycle is called one 'epoch'. Usually, NNs are trained for multiple epochs. Within one epoch, the training dataset can be divided into mini-batches, which will be passed through to the NN at one time. The number of data points that a mini-batch contains is called the 'mini-batch size'.

While parameters can be estimated by algorithms from the training data, 'hyperparameters' cannot be estimated from the data and are usually set manually by the modeller before training. NNs have various hyperparameters (like the number of layers and neurons). They may interact with each other in non-linear ways. Hyperparameter tuning is a procedure of finding the optimal hyperparameters of an NN (or other machine learning models), introduced in detail in Section 3.

2.2 | Different architectures of NNs used in surrogate modelling

Multilayer perceptrons have been widely used as surrogates in diverse disciplines (Roman et al., 2020). It has been shown that an MLP of one hidden layer (i.e., a shallow NN) with an adequate number of neurons can be trained to approximate any measurable function to any desired degree of accuracy (Hornik et al., 1989). As a result, studies using shallow MLPs are common in surrogate modelling. For example, Carnevale et al. (2012) use a one-hidden-layer MLP to learn the relationship between emissions and air quality indices. In the review by Razavi et al. (2012) of surrogate models in water resource modelling, 13 out of 14 papers used shallow NNs. However, deep NNs (i.e., with more than one hidden layer) might require fewer neurons to capture a similar level of complexity and thus are also applied as surrogates. For instance, Liong et al. (2001) use an NN with three hidden layers to mimic a hydrological model.

The second common type of NNs used as surrogate models is convolutional neural networks (CNNs) (LeCun et al., 1990), originally designed for image data. In contrast to MLPs where the neurons in one layer are connected to all neurons of the previous layer, CNNs use so-called 'convolution kernels' that slide across the input. Each neuron thus depends only on a local neighbourhood of neurons and not on all neurons of the previous layer. CNNs are also promising in handling time-series data (Fawaz et al., 2019). Although deeper CNNs might be able to capture more complex relationships, classical CNNs do not perform well as they grow deeper due to the problem of vanishing gradient (i.e., the gradients of the loss function approach zero, making NNs hard to train) (Bengio et al., 1994). To overcome this issue, residual networks (ResNets) (He et al., 2016) allow 'skip connections' to enable the training of deeper networks. An additional skip connection skips multiple layers of a neural network such that the output of one layer is not only fed to the next layer but also to the target layer of the skip

connection. Weber et al. (2019) find ResNets perform better than classical CNNs in surrogate modelling for climate forecasts.

The third common type of NNs used is recurrent neural networks (RNNs) (Elman, 1990; Rumelhart et al., 1986; Werbos, 1988), designed for sequence prediction tasks, such as speech recognition (Graves et al., 2013) and time series modelling (Hsu, 2017). RNNs are suitable for processing sequential data since the recurrent layers feed the output of the layer back to the layer itself such that the current state of a layer depends on the current input of the sequence as well as on the previous states of the layer. However, as the length of inputs increases, long-term dependencies are difficult to capture by classical RNNs (Marhon et al., 2013). Long short-term memory (LSTM) (Hochreiter & Schmidhuber, 1997) is a special RNN, capable of learning long-term dependencies. Rahmani et al. (2021) have developed LSTMs as surrogates of a process-based model to predict stream water temperature. LSTMs have been used to predict crop yields (e.g. Sun et al., 2019; Tian et al., 2021), but to our knowledge, they are not yet applied as surrogates of agricultural models. The BiLSTM (bidirectional long short-term memory) (Graves et al., 2005) is an extension of LSTM. It learns the sequence and the reversed sequence of the inputs. Alibabaei et al. (2021) use a BiLSTM to model evapotranspiration and soil water content in irrigation scheduling. RNNs are also helpful for non-sequential data. For example, Chopra et al. (2017) train an RNN with non-sequential data to predict whether a patient would be readmitted to the hospital.

Although MLPs have been applied as surrogates by Audsley et al. (2008) and Nguyen et al. (2019) (see Section 1), and LSTMs have been used to predict crop yields as mentioned above, using NNs of different architectures as surrogates is unexplored in agricultural models. Besides, no NN applications to approximate economic farm models are known to us. Given these research gaps, we employ the four different architectures of NNs including MLP, ResNet, LSTM and BiLSTM to develop surrogates of the detailed farm-level model FarmDyn.

3 | METHOD AND DATA

Our research design is shown in Figure 2. First, from the underlying farm model FarmDyn, we generate the data that will be used for training NNs. This involves defining the inputs/outputs of the farm model, generating data, and some data preparation steps. Second, for each of the four NN architectures, we define three different implementations that vary in depth (i.e., the number of layers). This results in 12 variants of depth, for which we optimise the remaining hyperparameters. The loss function used to train NNs is the MSE loss. It should be noted that minimising MSE is by construction equivalent to maximising R^2 (see Equation 2). We then select one best model in terms of R^2 from each variant of depth (in total 12 best models) and compare their inference time. Third, from each NN architecture, we select the best model with the most promising hyperparameters and inspect model performance in greater detail. Specifically, we examine model performance across varying amounts of training data by considering a set of evaluation metrics. The details of these three steps are described below.

3.1 | The underlying model and data generation

3.1.1 | Define inputs/outputs of the farm model

The bio-economic farm-level model FarmDyn covers a wide range of farm branches, such as arable, dairy, beef cattle, pig fattening and biogas. We focus on the arable farming branch. However, as a robustness check of the surrogate modelling pipeline, Appendix S1 presents

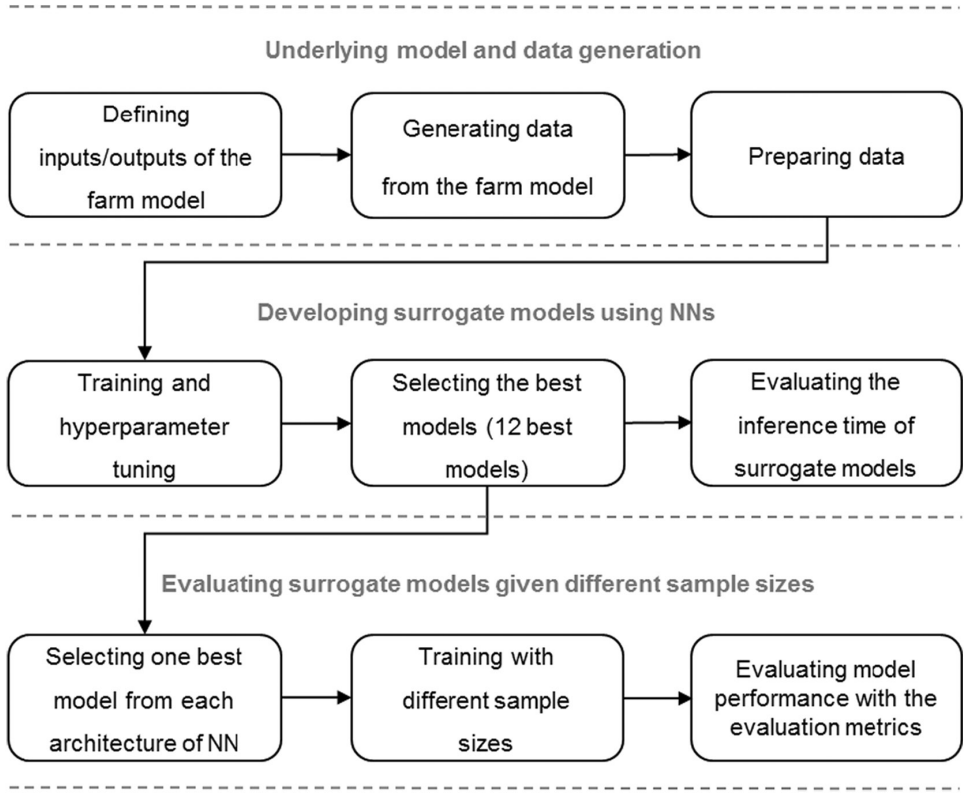

FIGURE 2 The overall research design of this study.

an additional smaller experiment for dairy farms in FarmDyn. In general, arable farms in FarmDyn make management decisions by maximising the net profit using MIP. In the version we use for this paper, the programming problem is constructed with several modules, with each module dealing with a specific aspect of a farm. For more details, see the documentation of FarmDyn: https://farmdyn.github.io/documentation/.

1. Economic module: this defines the economic aspects of the farm-level model, including objective function (net profit maximisation), cash flow structure, income tax calculation, premium payments, sales and production levels, variable cost structure, and investment costs.
2. General cropping module: this optimises the cropping decisions subject to land availability, yields, maximal crop rotational shares, crop prices, machinery and fertiliser needs, and other variable costs of crops.
3. Labour module: this optimises labour allocation for different detailed on- and off-farm activities with a monthly resolution.
4. Environmental accounting: this quantifies farm-level methane (CH_4), ammonia (NH_3), nitrous dioxide (N_2O), nitrogen oxides (NO_x) and elemental nitrogen (N_2), as well as particulate matter formation (PM10 and PM2.5).
5. Fertilisation ordinance: this adds the fertilisation constraints based on the German implementation of the Nitrates Directive, including nutrient balance restrictions, threshold for organic nitrogen application quantities, restriction of fertiliser application in autumn, and others.

6. Greening: this captures the greening requirements of the Common Agricultural Policy of the European Union (CAP) as applicable since 2013. It integrates the key measures of crop diversification and ecological focus areas (mainly legumes) into FarmDyn.

To build a surrogate model of FarmDyn, it is necessary to define the model interface clearly. This means we need to define what input variables we pass to the model and what output variables we aim to obtain. In our case, the surrogate model takes the same inputs and produces the same outputs as the underlying model FarmDyn. Therefore, defining the inputs/outputs of FarmDyn will technically define the inputs/outputs of the surrogate model.

Table 1 summarises the inputs and outputs of arable farms in FarmDyn. They include variables about crops, farming inputs, machinery, farm endowment, environmental indicators, and farm accounting. Crops included in the model are winter wheat, winter barley, winter rapeseed, summer cereal, maize and sugar beet. The farming inputs include diesel, fertiliser (urea-ammonium nitrate, phosphorus and potassium), seed, lime, herbicide, fungicide, insecticide, growth control, water, and hail insurance. In total, there are 77 inputs and 248 outputs. There are many constant parameters in FarmDyn, but we exclude them here since the surrogate model should be able to learn the underlying constant parameters that reflect the relationship between inputs and outputs. The detailed lists of inputs and outputs can be found in Appendix S1.

We develop a surrogate model that predicts 248 outputs simultaneously instead of building one separate model for each output. The advantages of this approach are: (1) it scales better with the number of outputs and is less time-consuming than training many separate models; and (2) it is easier to integrate only one surrogate model into the future ABM than many small ones. However, training separate models for each output makes it easier to observe the loss function of each output, thus it could be easier to improve the model accuracy for some particular outputs. Nonetheless, when training a neural network with multiple outputs, one can weight them differently, then the loss function will react more to those 'more important' outputs. In this paper, we treat all outputs equally since we do not target any specific application here.

3.1.2 | Data generation and preparation

The initial farm data is generated from FarmDyn by Latin Hypercube Sampling (LHS) (McKay et al., 1979). LHS independently stratifies each input dimension into N equal intervals, where N is the number of data points. For a given dimension, it generates one data point in each interval and randomly combines this with the selected interval of the other dimensions. LHS provides outcomes from a uniform distribution of the data within the design space (Tyan & Lee, 2019).

The optimal amount of data used to train a surrogate model depends on the complexity of the problem and the computational budget available. Since NNs need large datasets for problems with high dimensionality, we generated as many data points as possible given our time budget. With 10,000 model outcomes (i.e., observations) each time, the data generation process ran 17 times and produced 163,480 data points (taking about 45 h) because FarmDyn did not successfully solve for some input draws due to implausible input combinations.

The whole dataset is then randomly split into two subsets including the training set (90%) and the test set (10%), having 147,132 and 16,348 observations, respectively. The training set is used to train the model, and a test set is solely used to assess the model. During the training process, 10% of the training set is used as a validation set to monitor the models' performance on unseen data to avoid overfitting, meaning the network learns too much information that is specific to the training data and does not generalise for other datasets. The validation set is

TABLE 1 Summary of inputs and outputs of arable farms in FarmDyn.

	Inputs (unit)	Outputs (unit)
Crops	• Selling price of crops (€/t)	• Production level (ha) • Production quantity (t) • Sale quantity (t) • Crop revenues (€) • Amount of fertiliser used (kg/ha) • Output quantity of crop residues (t) • Revenue from crop residues (€)
Farming inputs	• Price (€/L, €/kg, €/t, €/ha)	• Used amount (L, kg, t, ha) • Cost (€)
Machinery	• Price (€/unit)	• Applied area (ha) • Fixed cost (€) • Variable cost (€)
Farm endowment	• Farm size (ha)	• Amount of idle land (ha) • Shadow price of land (€/ha) • Distribution of labour to each month (hours) • Distribution of labour to on-farm and off-farm work opportunities (hours)
Environmental indicators	• Nitrogen needed from mineral fertiliser per ha (kg/ha) • Phosphate need from mineral fertiliser per ha (kg/ha)	• Average nitrogen/phosphate input (kg/ha) • Average nitrogen/phosphate surplus (kg/ha) • Nitrogen leaching on the farm (kg) • Phosphorus loss on the farm (kg) • Emissions on the farm (e.g. phosphorus, nitrous oxide, nitrate, CO_2 equivalent, particulate) (kg) • Global warming potential as CO_2 equivalent (kg) • Particulate matter formation potential (kg) • Terrestrial acidification potential (kg) • Freshwater eutrophication potential (kg) • Marine water eutrophication potential (kg)
Farm accounting	/	• Variable costs of crops, fertilisers, phytosanitary, machinery (€) • Total crop revenue (€) • Off-farm income (€) • Sum of investments (€) • Profit (€) • Cash flows (€) • Withdraw (€) • Depreciation (€) • Total premium (€) • Income (€)
Total number of variables	77	248

also used to implement 'early stopping' given a stopping criterion; for example, when the loss function stops decreasing after a certain number of epochs. The difference between the validation set and the test set is that the test set is used to assess the final performance of a trained model, but the validation set is a part of the training set that is used during training to monitor the performance of the model.

Normalisation of data is recommended since it usually leads to faster convergence (Huang et al., 2020). For the training set, we normalised both input and output data between 0 and 1 with the 'MinMaxScalar' of the python package 'scikit-learn' (Pedregosa et al., 2011). The test set is then normalised by the scalar of the training set because the unseen data must fit into the trained scale of the NN. This means that when later integrating the surrogate model into an

ABM, the input data must also be normalised before being used in the surrogate model. This can be achieved by either ABM modellers or surrogate modellers by adding a normalisation layer to the surrogate model.

It is worth noting that MLP can directly receive the one-dimensional (1D) data as input. In our case, an MLP as a surrogate model of FarmDyn should have 77 neurons in the input layer and 248 neurons in the output layer as shown in Figure 1. For CNNs, we use a 1D convolutional kernel for the 1D data. Both 1D and 2D convolutional kernels are commonly used in deep learning and are implemented by popular deep learning libraries, such as Keras (Chollet, 2015) and pytorch (Paszke et al., 2019). In this paper, we use *Conv1D* layer in Keras to process 1D data. RNNs treat 1D inputs as sequential data. In our case, the 77 input variables are treated as 77 time steps.

3.2 | Developing surrogate models using NNs

3.2.1 | Training and hyperparameter tuning

Developing surrogate models using NNs means obtaining well-fitted NNs that can approximate the underlying model. As mentioned above, while the training process that optimises parameters of an NN is automatically done by computer algorithms, hyperparameters are usually set by modellers manually before training. Types and numbers of hyperparameters differ among different types of NNs. Here, we focus on the main hyperparameters.

The number of hidden layers (i.e., the depth) is a determinant for NNs' ability to capture complex relationships. Although deep networks might perform better than shallow networks, increasing the depth does not always improve the performance (He et al., 2016). To compare the performance of NNs of different depths, we train three variants of depth for each architecture of NN: for MLP, LSTM and BiLSTM, we train models with one, two and three hidden layers, respectively; for ResNet, we train models with 18, 34 and 50 layers, as these are proposed by the original paper of He et al. (2016). Our ResNets are 1D CNNs due to the characteristics of our input data. The hyperparameter tuning for the 12 variants was done according to the following steps (Table 2).

Step 1: Number of neurons in a hidden layer/Number of filters in the second stage

For each variant of depth of MLP, LSTM and BiLSTM, we must tune the number of neurons in each hidden layer since it is an important hyperparameter determining the performance of NNs. A small number of neurons could lead to underfitting, meaning the network is not complex enough to capture underlying relationships in the data. A high number could cause overfitting. We experiment with the number of neurons of {32, 64, 128, 256, 512, 1024, 2048} in each hidden layer.

For the three variants of ResNets, we tune the number of filters in the second stage. The commonly used ResNets have five stages of convolutional process. The number of filters in the second stage will automatically determine the number of filters in the following stages (He et al., 2016). The training process of ResNets estimates the weights of all filters. We explore the space of {16, 32, 64, 128, 256, 512} for this hyperparameter.

Step 2: Learning rate

The learning rate determines the speed of the algorithm to head to the next solution in the parameter search space. A small learning rate takes a long time for the network to converge, and a large learning rate might cause the network not to converge. We explore the space of {0.0001, 0.0003, 0.001, 0.003, 0.01} for learning rate for all models.

TABLE 2 Steps and search spaces of hyperparameter tuning of different NNs.

Steps	Tuned hyperparameter at this step	MLP with hidden layer(s) of {1, 2, 3}	LSTM with hidden layer(s) of {1, 2, 3}	BiLSTM with hidden layer(s) of {1, 2, 3}	ResNet with layer(s) of {18, 34, 50}	The default setting of other hyperparameters at this step
Step 1	Number of neurons in each hidden layer	{32, 64, 128, 256, 512, 1024, 2048}			/	Learning rate = 0.001 Mini-batch size = 32 Optimiser = Adam Activation function = ReLu (for MLP and ResNet), tanh (for LSTM and BiLSTM) epochs = 200 (early stopping)
	Number of filters in the second stage	/			{16, 32, 64, 128, 256, 512}	
Step 2	Learning rate	{0.0001, 0.0003, 0.001, 0.003, 0.01} for all NNs				Mini-batch size = 32 Optimiser = Adam Activation function = ReLu (for MLP and ResNet), tanh (for LSTM and BiLSTM) epochs = 200 (early stopping)
Step 3	Mini-batch size	{16, 32, 64, 128} for all NNs				Optimiser = Adam Activation function = ReLu (for MLP and ResNet), tanh (for LSTM and BiLSTM) epochs = 200 (early stopping)
Step 4	Optimiser	{Adam, Adamax, RMSprop, SGD} for all NNs				Activation function = ReLu (for MLP and ResNet), tanh (for LSTM and BiLSTM) Epochs = 200 (early stopping)

Step 3: Mini-batch size

Mini-batch size determines how often the loss function is computed in one epoch and thus influences the updates of parameters. We experiment with a mini-batch size of {16, 32, 64, 128} for all models.

Step 4: Optimiser

Optimisers determine how the parameters of a network are changed to reduce the loss function. We experiment with {Adam (adaptive moment estimation), Adamax (a variant of Adam based on the infinity norm), RMSprop (root mean squared propagation), SGD (stochastic gradient descent)}.[2]

Since we only tune one hyperparameter at each step, the rest of the hyperparameters should be set with default values in order to start training. The last column of Table 2 indicates the default setting of other hyperparameters at each step besides the tuned hyperparameter. As can be seen, we do not tune the activation function. For MLP and ResNet, we use the ReLU activation function; for LSTM and BiLSTM, we use the tanh activation function (i.e., hyperbolic tangent function) to enable faster training on the GPU (graphics processing unit). Early stopping is used to determine when the training process should be stopped. The maximum number of epochs is set to 200, but the training process will be terminated when the validation error stops decreasing after 15 epochs. The performance of an NN is recorded after each epoch, and the model with the lowest MSE on the validation set will be saved as the trained model. All NNs are built and trained using the 'Keras' library (Chollet, 2015). To run the experiments, we use an 11 GB GPU (NVIDIA GeForce RTX 2080 Ti).

3.2.2 | Selecting the best models and evaluating the inference time

After hyperparameter tuning, we select the 12 best models (three variants of depth from each architecture) according to the R^2 on the test set. Then, we examine the inference time, defined as the time to make one prediction (i.e., one forward run of the NN), of the selected models. Inference time determines the efficiency of the surrogate model in future applications. To make a fair comparison between the trained NNs and FarmDyn, we record the simulation time of FarmDyn and the inference time of NNs on the same machine (with Intel Xeon CPU E5-2699 V4, 2.20GHz). The same experiment is repeated five times, and the average inference time per data point of each NN is calculated to avoid fluctuations in computing time.

3.3 | Evaluating surrogate models

3.3.1 | Motivation to evaluate surrogate models beyond R^2

The training process described above uses the entire training set we have generated. However, from a practical perspective, there are two aspects we must consider when developing and applying the surrogate model: (1) Generating data from highly detailed farm-level models could be time-consuming. Although increasing the amount of data is always possible, it is time-consuming and computationally expensive for modellers. From a practitioner's perspective, a natural question is how to determine when to stop generating data. For this, we need to assess how additional data can affect model performance, and we need to determine when the surrogate model has obtained an accuracy that is sufficient for the envisioned application. The latter is usually difficult to assess based on the R^2 measure; (2) Different model architectures might differ substantially in inference time, with MLPs being much faster than the other architectures and possibly only minor differences in model performance in terms of R^2. Thus, we need

additional evaluation metrics that are more targeted to the application of the surrogate model to judge if those differences in model performance are relevant from a practical perspective and justify a longer inference time.

Therefore, we first develop evaluation metrics besides R^2 (see Section 3.3.2 and Table 3) that are relevant for the application of the surrogate model. Secondly, we perform a simulation exercise (see Section 3.3.3) where we evaluate the performance of the four different architectures when trained with varying amounts of data. These simulations allow us to assess if similar performance could be achieved with a smaller size of training set or if we can expect performance increases from additional data. Additionally, it allows us to inspect how the alternative model evaluation criteria (besides R^2) behave when varying the size of the training set. This helps us decide the amount of data required to achieve acceptable performance from an application point of view.

3.3.2 | Evaluation metrics

1. **Goodness of fit**

Like many other studies (see Roman et al., 2020), we first use R^2 to measure the goodness of fit of a surrogate model on the test set. R^2 measures the proportion of the total variation in an output variable explained by the model. For an output variable y, the R^2 in terms of this output is calculated with Equation (2).

$$R^2_y = 1 - \frac{\sum (y_i - \hat{y}_i)^2}{\sum (y_i - \bar{y})^2} \tag{2}$$

where y_i is the simulated value (i.e., 'true' value generated by FarmDyn in our case) of the output y of the observation i, $\hat{y}_i$ is the predicted value of the output y of the observation i, and $\bar{y}$ is the mean of the simulated values of the output y.

The R^2 typically ranges from 0 to 1. However, when a model's performance is worse than simply predicting the mean of the output for all observations, it becomes negative.[3] Since we only select models that have a positive R^2, it ranges from 0 to 1 in this study. We calculate the average R^2 across all outputs with Equation (3):

$$R^2 = \frac{1}{K} \sum R^2_{y_k} \tag{3}$$

where K is the number of outputs ($K = 248$ in this study).

Besides R^2 measure, we also include root mean squared error (RMSE) as another goodness of fit measure. It can reflect the prediction error of outliers well by squaring the difference between the simulated and predicted values. For an output y, RMSE in terms of this output is calculated with Equation (4):

$$RMSE_y = \sqrt{\frac{\sum (y_i - \hat{y}_i)^2}{N}}, \tag{4}$$

where y_i is the simulated value of the output y of the observation i, $\hat{y}_i$ is the predicted value of the output y of the observation i, and N is the number of observations in the test set. We then calculate the average RMSE across all outputs with Equation (5):

TABLE 3 Summary of the evaluation metrics.

Criterion	Example measurement	Notation	Range
Goodness of fit	Average R^2 across all outputs	R^2	(0, 1)
	Average root mean squared error	RMSE	(0, + ∞)
Consistency of bivariate relationships	Average APE between simulated and predicted MICs between selected variables: 1. The amount of N applied and N leaching on a farm 2. Price of sugar beet and share of sugar beet revenue in total farm revenue	$APE_{relationship}$	(0, + ∞)
Accuracy of capturing corner solutions	Accuracy in capturing corner solutions of crop choices	$Accuracy_{corner}$	(0, 1)
Accuracy in holding constraints	Accuracy in holding the constraint of farm size	$Accuracy_{constraint}$	(0, 1)

$$RMSE = \frac{1}{K} \sum RMSE_{y_k} \tag{5}$$

where K is the number of outputs (K=248 in this study).

2. **Consistency of bivariate relationships**

From an application perspective, it is crucial to assess whether some fundamental relationships between an input and an output or between two outputs are learnt by the surrogate model. This is particularly important for applications where the results of scenarios strongly depend on the relationship between certain variables. Here, two examples demonstrate how the consistency of bivariate relationships can be evaluated. The first one is from the environmental perspective. When simulating scenarios about nitrogen (N) fertilisation and its environmental impact, modellers would want to check if the surrogate model can correctly capture the relationship between fertilisation decision and the environmental outcomes. Here, we consider if the relationship between the amount of N applied and the amount of N leaching on a farm is learnt by the surrogate model. In FarmDyn, for a specific crop of a certain yield level, the relationship between the two variables is linear. However, it becomes non-linear at the farm level due to different combinations of crop choices. The second example is from the economic perspective. When the selling price of a crop increases, farms tend to cultivate more of this crop to maximise the profit. Since we look at the relationship between crop price and crop revenue across farms that have different farm endowments (e.g., farm size), it is more meaningful to look at the relationship between crop price and the share of this crop revenue in total farm revenue, rather than the relationship between crop price and crop revenue at farm level. Here, we use the example of sugar beet price and share of sugar beet revenue in total farm revenue, which is also not linear according to the simulated data.

To capture the non-linear relationship between two variables, we use the maximum information coefficient (MIC) (Reshef et al., 2011, see Appendix S1), a non-parametric method that has been widely applied (Cao et al., 2021). MIC ranges from 0 to 1. The higher the MIC is, the stronger the relationship between the two variables. We use the python package 'minepy' (Albanese et al., 2013) to calculate the MICs. Then, we calculate the absolute percentage error (APE) between the simulated and predicted MICs between the two variables with Equation (6):

$$APE_{relationship} = | MIC_{simulated} - MIC_{predicted} | / MIC_{simulated} \tag{6}$$

where $MIC_{simulated}$ is the MIC between the two variables calculated from the simulation data, and $MIC_{predicted}$ is calculated from the predicted data of the surrogate model.

The average $APE_{relationship}$ of the above-mentioned two groups of variables is calculated at the end.

3. Accuracy in capturing corner solutions

Another important aspect for the application of surrogate models is its ability to capture corner solutions. These are special solutions to an optimisation problem in which the quantity of one of the arguments in the objective function is zero (Debertin, 2012). In arable farming, examples of corner solutions are an available technology that is not chosen or a particular crop that is not produced. A previous study has shown that capturing corner solutions is usually challenging for surrogate models (Seidel & Britz, 2019). The ability of the model to capture corner solutions is difficult to assess from R^2. It would be interesting to see if the surrogate model is able to capture corner solutions, that is, if it at least gets the farmers' basic crop choices correct without considering the level. This dimension becomes particularly relevant if farmers' choices are the focus of the analysis in applying surrogate models, for example when simulating farmers' technology adoption decisions.

For example, we measure NNs' ability to capture corner solutions of farmers' crop choices. For a crop c, we first transform its simulated and predicted production levels for each observation into binary: 0 (if not produced[4]) and 1 (if produced). Then, we count the number of farms whose decisions are correctly predicted. The accuracy in capturing corner solutions of crop c is calculated with Equation (7):

$$A_c = \frac{1}{N} a_c \tag{7}$$

where a_c is the number of observations whose decision on crop c is correctly predicted, and N is the number of observations in the test set.

The average accuracy in capturing corner solutions across all crops is calculated with Equation (8):

$$Accuracy_{corner} = \frac{1}{C} \sum A_c \tag{8}$$

where C is the number of crop types ($C=6$ in this study).

4. Accuracy in holding constraints

Individual farm optimisation models simulate farmers' choices to maximise an output subject to a set of constraints (e.g., land/labour endowment). When employing a surrogate model of such an individual farm model, it is crucial that those constraints hold. For example, the sum of the planted areas of all farm crops cannot exceed the farm size if renting land is impossible. From an economic modelling point of view, a smaller violation of these constraints by the surrogate model is often more problematic than a larger deviation from the underlying model behaviour within the feasible solution space (e.g., some underutilisation of a resource). R^2 does not capture this, as it does not distinguish between feasible and infeasible solution space given by the constraints of the underlying model. Therefore, a dedicated measure of how well the prediction of the surrogate model obeys the constraints is warranted.

As an example, we measure NNs' accuracy in holding constraints of farm size with Equation (9):

$$Accuracy_{constraint} = \frac{1}{N} a_{constraint} \tag{9}$$

where $a_{constraint}$ is the number of observations whose constraints of farm size are not violated, and N is the number of observations in the test set.

3.3.3 | Training with different amounts of data

To investigate the impact of the amount of training data on the performance of the surrogate model, we choose the best model with the most promising hyperparameters from each NN architecture and train them with varying amounts of training data. We split the original training set (Section 3.1.2) into sizes of {1000, 5000, 10,000, 50,000, 100,000, 147,132[5]}. The test set is the same as before, containing 16,348 data points, but it is normalised according to the scale of each training set. To avoid fluctuations, we average the performances of five models trained with the same data using different random seeds for each architecture of NN and for each size of training set.

4 | RESULTS AND DISCUSSION

4.1 | The best models and their inference time

We select the 12 best models in total (three variants of depth from each architecture) in terms of R^2 on the test set. Table 4 shows the architecture of the selected NNs. As can be seen, BiLSTM3 (BiLSTM with three hidden layers) has the highest R^2 of 0.99, while ResNet18 has the lowest R^2 of 0.93. This shows NNs can capture the variance in the data very well. In terms of R^2, we observe that BiLSTMs and LSTMs perform better than MLPs and ResNets. RNNs, although designed for sequential data, can also adapt to non-sequential data. Appendix S1 provides the detailed scatter plots of the predictions of BiLSTM3 (the model with the highest R^2) and simulated results of a few outputs that are usually important in applications.

As shown in Figure 3, the inference time of different NNs differs substantially. MLPs are the fastest in predicting, whereas LSTMs and BiLSTMs are much slower, reflecting the larger number of parameters than MLPs (see Table 4). FarmDyn takes 5.40 s to generate one data point on average. In comparison, the MLP3 (MLP with three hidden layers) ($R^2 = 0.95$) needs 0.000026 s to predict one data point being about 207,000 times faster than FarmDyn, and the BiLSTM3 ($R^2 = 0.99$) takes 0.021 s, being 257 times faster. Whether this speed is satisfying depends on the time budget of future applications.

4.2 | Model performance and impact of the amount of training data

According to Table 4, we select the four best model specifications in terms of R^2 to experiment with different sizes of training set as described in Section 3.3.3. They are MLP with 2 hidden layers (MLP2), ResNet with 50 layers (ResNet50), LSTM with 3 hidden layers (LSTM3), and BiLSTM with 3 hidden layers (BiLSTM3). In the following, we refer to them as MLP, ResNet, LSTM and BiLSTM without repeating the number of layers.

4.2.1 | Goodness of fit

Figure 4a,b show the change of R^2 and RMSE (calculated using the normalised data) of the selected NNs with varying amounts of training data. With a training set of 1000 observations, BiLSTM and MLP can achieve an average R^2 of 0.8, whereas LSTM can only achieve around 0.55. For ResNet, 1000 observations for training are insufficient to converge because the R^2 of ResNet trained with this amount of data is negative (not shown in the figure).[6] As the size of training set increases from 1000 to 5000, we see a steep increase in R^2 for all four types of models. With 50,000 data points for training, BiLSTM and MLP can already achieve a R^2 of

TABLE 4 The architectures of the 12 selected models based on R^2 on the test set.

	Number of hidden layers	Number of neurons in each hidden layer	Number of filters in the second stage	Learning rate	Mini-batch size	Optimiser	Number of parameters	R^2
MLP1	1	128	/	0.001	32	RMSprop	41,976	0.94
MLP2	2	64, 512	/	0.0003	32	Adam	165,496	0.96
MLP3	3	128, 32, 256	/	0.0003	32	Adam	86,296	0.95
ResNet18	18	/	32	0.001	128	Adam	1,119,960	0.93
ResNet34	34	/	8	0.0003	32	Adam	171,648	0.94
ResNet50	50	/	16	0.001	64	Adam	1,666,648	0.94
LSTM1	1	256	/	0.001	32	Adam	327,928	0.97
LSTM2	2	128, 64	/	0.001	32	Adam	132,088	0.97
LSTM3	3	32, 128, 1024	/	0.001	32	Adam	5,063,672	0.98
BiLSTM1	1	2048	/	0.001	32	Adamax	34,603,256	0.98
BiLSTM2	2	32, 256	/	0.001	32	Adamax	793,336	0.98
BiLSTM3	3	32, 128, 512	/	0.001	32	Adamax	3,610,360	0.99

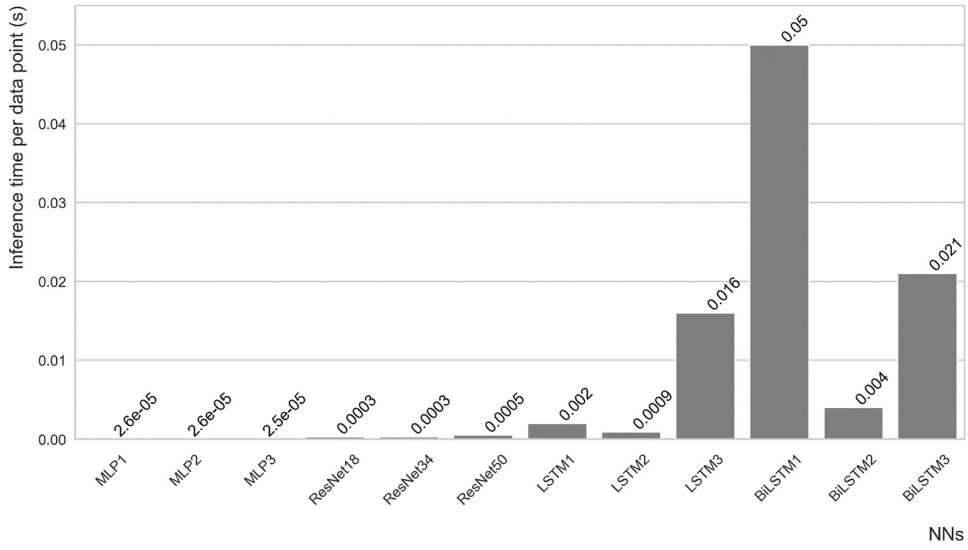

FIGURE 3 Inference time per data point of each NN.

around 0.95. Interestingly, with 100,000 data points, all models except for LSTM are already close to their maximum performance level, where additional data is of little benefit.

4.2.2 | Consistency of bivariate relationships

Figure 4c shows the measure for the ability to capture the relationships of two groups of variables as mentioned above. With more and more training data, the $APE_{relationship}$ of BiLSTM goes down steadily, achieving an APE of 0.70% with 100,000 observations. In comparison, MLP can also reach a similar level of accuracy but with fluctuations when the size of training set is smaller. BiLSTM, MLP and LSTM all achieved the best performance in capturing the relationships with 100,000 observations, while ResNet has a much higher level of error of 8.35% given the same amount of training data.

4.2.3 | Accuracy in capturing corner solutions

Figure 4d shows the accuracy in capturing corner solutions of crop choices ($Accuracy_{corner}$) of each NN architecture trained with different amounts of data. With 10,000 data points for training, BiLSTM can achieve accuracy near to 100% in capturing the corner solutions of crop choices. Once the size of training set exceeds 50,000, the accuracy does not increase much for most models except for LSTM. We can also see that MLP is as good as BiLSTM in capturing corner solutions at and beyond 50,000 data points.

4.2.4 | Accuracy in holding constraints

Figure 4e shows the accuracy of NNs in holding constraints of farm size ($Accuracy_{constraint}$). With a smaller training set (less than 20,000 data points), MLP outperforms BiLSTM with an accuracy of 0.98, but BiLSTM dominates once the size of training set reaches 50,000. Furthermore, the

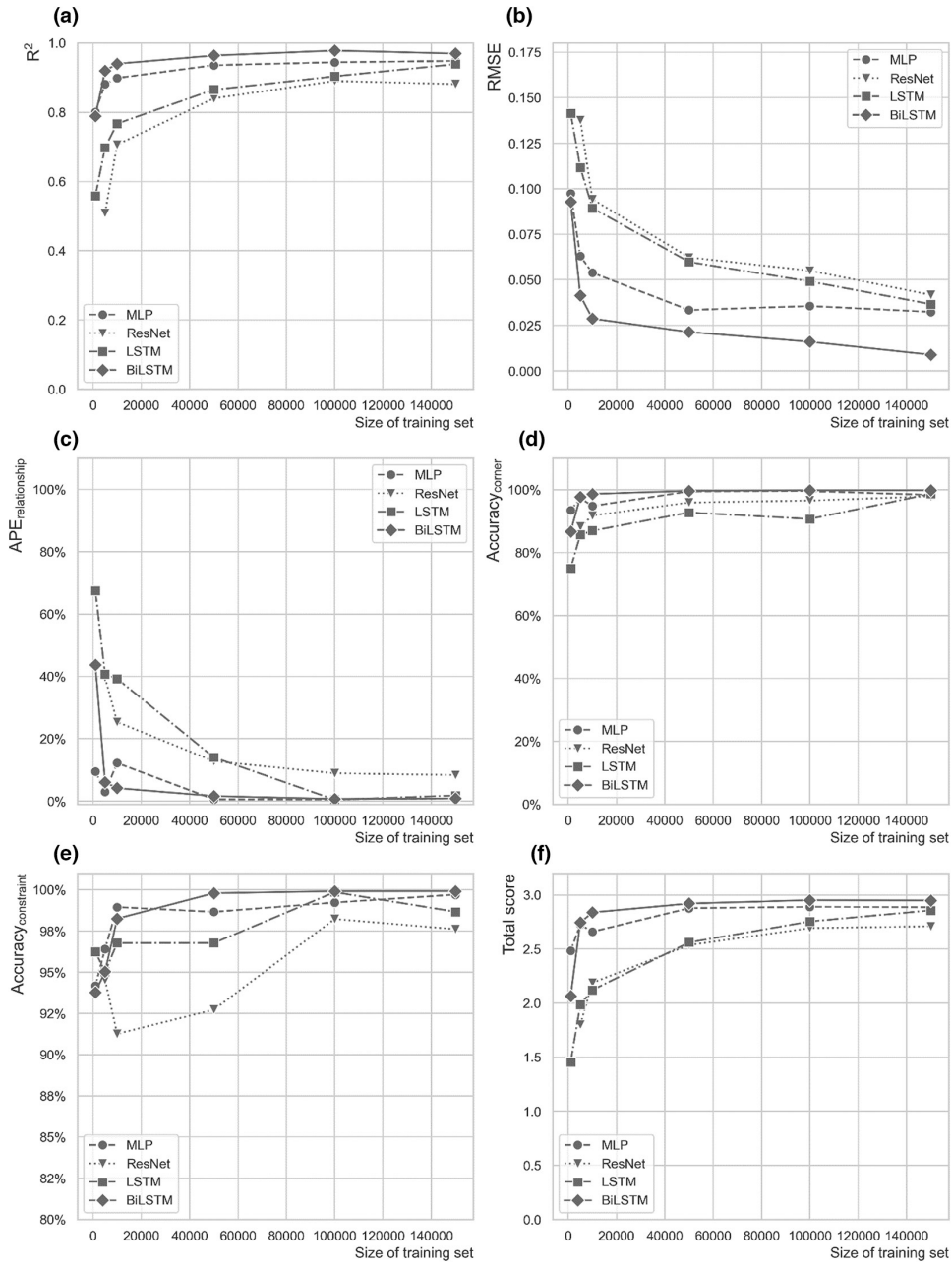

FIGURE 4 Performance of different architectures of NNs given different sizes of training set.

accuracy of BiLSTM in holding the constraints is very close to 100%, given 50,000 data points. After this point, adding more data points does not improve the performance of BiLSTM.

Figure 4f shows the total score of each NN, which is calculated by simple addition and subtraction of all criteria ($Total\ score = R^2 - RMSE - APE_{relationship} + Accuracy_{corner} + Accuracy_{constraint}$) because they all were chosen to be in the range of 0 and 1 in this study. As can be seen, increasing the size of training set from 1000 to 50,000 significantly improves the performance of all types of models. Once the size of training set reaches 100,000, adding more observations to the training process does not necessarily improve the performance of surrogate models. Thus, in our case, a

size of training set between 50,000 to 100,000 should be sufficient to develop surrogate models that perform well concerning all our evaluation metrics. In terms of model preferability, BiLSTM almost always dominates over other types of NNs given different amount of training data but has a close competitor—MLP. Considering the inference time of the trained model, MLP may be the go-to model in many surrogate model applications that require a large number of model runs.

The surrogate models developed by Troost et al. (2022) capture the underlying relationship between 22 inputs (prices and model uncertainty parameters) and 9 outputs (crop areas). When the size of the training sample is below 1000 data points, the performance of the surrogate models increases the most. With many more input and output variables in our case, training surrogate models requires more data. In terms of inference time, deep learning methods used in our paper are 257–207,000 times faster than the detailed farm-level model FarmDyn, whereas the surrogate models in Troost et al. (2022) are 1800–60,000 times faster than their underlying farm model.

5 | SURROGATE MODELS FOR AGRICULTURAL POLICY SIMULATION IN AN ABM: A CONCEPTUAL DISCUSSION

The previous sections show that developing a surrogate model of a farm-level model like FarmDyn is possible and they provide practical guidance to do so. Here, we discuss some implications of how such a surrogate model can be used for agricultural policy simulation in an ABM, as well as further avenues opened up by it. Additionally, we comment on the challenges and potential downsides when using surrogate models.

Integrating a surrogate model of a farm-level model into an ABM makes it possible to represent the decision-making mechanism of agents (i.e., the behaviour of the underlying individual farm-level model) with the surrogate model, whereas the landscape of the selected region and interaction rules among agents are determined by the ABM. Prior to any simulation, the ABM initialises the farm population for the selected research region. This might include defining the types of farms and the original number of farms that belong to each farm type, which reflects the characteristics of the farm population in the research region. In the case of AgriPoliS, farms are initialised and differentiated from each other in terms of location, farm size, equity, availability of labour, existing capacity of machinery, age of the farm operator, and so on. Some of these farm characteristics serve as an input to the surrogate model for determining agents' behaviour. The ABM keeps track of the actions of each agent and their interactions and updates farm characteristics for the next period accordingly. For example, if a farm has acquired additional land or new investment in one period, it needs to be accounted for in the next period.

As one of the main opportunities of using a surrogate model to couple complex farm-level models, such as FarmDyn, with ABMs, such as AgriPoliS, we consider the possibilities to simulate agri-environmental policy impacts. One of the strengths of FarmDyn is the comparably rich representations of bio-physical processes, farm technologies and farm management decisions. For example, FarmDyn captures the whole nitrogen flow (bio-physical processes) on the farm, which can be altered using low-emission manure techniques (farm technology) or exporting on-farm manure (farm management decision). FarmDyn also allows assessment of a wide range of environmental indicators (e.g., nitrogen balances). This enables us to simulate and assess outcomes of policies that limit fertiliser use in certain locations, reflecting on the one hand management decisions by individual farmers through FarmDyn, and on the other hand accounting for (spatially explicit) interactions between farms on the land market simulated in AgriPoliS. Eventually, the connection between both models allows us to assess spatial environmental policy effects based on FarmDyn's environmental indicators.

It is important to note that while surrogate models offer new possibilities in coupling complex individual farm-level model with ABMs, it is still bound by the capabilities of each individual

model. For example, the current version of FarmDyn, which is considered in this paper, does not allow farms to switch between types, for example turning from an arable farm to a dairy farm. Hence, this capability is also not included in the surrogate model. In Appendix S1, we present an additional surrogate modelling experiment which tests the applicability of the approach on a dairy farm type. If a switch between farm types is required from the ABM perspective, either FarmDyn needs to be extended by this feature, or a surrogate model needs to be trained on data of multiple farm types, which internally can determine the resulting farm type.

Beyond surrogate models for ABMs, an entirely different potential use of surrogate models is for more efficient calibration of the underlying model (Storm et al., 2020). Assume that we want to calibrate FarmDyn to some empirically observed crop shares, that is, we want to minimise the difference between the observed crop shares (calibration target) and the crop share output of FarmDyn. For the calibration case with FarmDyn, we could consider yields as calibration parameters; however, other technology parameters or prices are also conceivable (see Britz, 2021 for a detailed description of this calibration procedure). In the case of an NN-based surrogate model, one would then use yields as varying input variables in addition to the more general input variables such as farm endowments. In the next step, the surrogate model can learn new input/output relationships under different yield levels. For the calibration in FarmDyn, we can then use the trained surrogate model to find the yield level that minimises the difference between the FarmDyn output and the defined calibration target, which is in our example the crop shares. The potential advantage of using the NN-based surrogate model for calibration is that gradients (i.e., how outputs change in response to changes in inputs) can be calculated analytically and in a highly efficient manner. On the contrary, for the underlying model, gradients need to be calculated numerically, which is computationally expensive. In theory, this idea can be extended to calibrate the ABM coupled with a surrogate model. In this case, it would require building a surrogate model for the entire ABM. The surrogate model learns the relationship between the varying inputs (including parameters to be calibrated and the input variables of the ABM) and the ABM outputs. Parameters to be calibrated could be, for example, one that specifies interaction behaviour on the land market. Similar to the calibration of a farm-level model using surrogate models, the ABM parameters can be efficiently calibrated according to the gradients.

Despite the benefits of surrogate modelling, we must be aware of its limitations. First, we need to consider that although surrogate models themselves are computationally efficient, training surrogate models, especially hyperparameter tuning, is time-consuming and requires considerable computational resources (Troost et al., 2022). Second, deep-learning-based surrogate models are restricted in their validity to the range of input values in the training data. This means that once the ranges of input data are extended, surrogate models must be retrained. Retraining might also be necessary each time the underlying model is updated, either to consider new features or to resolve bugs, or if the model needs to be adjusted for a new study or research question. This frequent need for retraining might counteract the advantage of reusability of surrogate models. However, future research can overcome the difficulty by automating the training process as far as possible. It is also important to consider that it might not be necessary to repeat the entire hyperparameter search process, as long as the fundamental complexity of the model is not changed substantially. This makes retraining substantially less costly and automation more feasible.

6 | CONCLUSIONS

We investigate the performance of NNs of different architectures in approximating the behaviour of a detailed farm-level model FarmDyn. We compare the performances of four architectures of NNs (MLP, ResNet, LSTM and BiLSTM), considering 12 different implementations

in terms of model depth. The trained NNs are supposed to accurately map the relationship between 77 input variables and 248 output variables of the farm model. The high goodness of fit of the selected surrogate models shows that NNs can explain most of the variation in the output variables. The BiLSTM with three hidden layers achieves an average R^2 of 0.99 across all output variables, whereas the lowest average R^2 is 0.93 by ResNet with 18 layers. BiLSTM and LSTM achieve better performance than other types of NNs, although they are originally designed to handle sequential data. In terms of inference time, all trained NNs are much faster than FarmDyn. MLPs are about 207,000 times faster, and the best performing BiLSTM regarding R^2 is still 257 times faster.

We also provide generic evaluation metrics to assess the performance of surrogate models, which can offer future modellers additional help in selecting surrogate models in applied modelling. The evaluation metrics consist of four dimensions: (1) Goodness of fit; (2) Consistency of bivariate relationships; (3) Accuracy in capturing corner solutions; and (4) Accuracy in holding constraints. They are calculated for different sizes of training set used for training to understand the effort needed in data generation. In our specific case, increasing the size of training set from 1000 to 50,000 significantly improves the performance of all types of models. Once the amount of training data reaches 100,000, adding more data points for training does not improve the performance of the surrogate models in any relevant way as defined by the evaluation metrics. MLP performs the second best in general, and its performance on other criteria is close to the best model—BiLSTM. Since it has a strong advantage on inference time, MLP might be the prime choice for many cases with strong computational demands.

Our research shows NNs are efficient in approximating detailed farm-level models. Thus, they can offer upscaling possibilities of ABMs with detailed farm-level model outcomes. Specifically, the integrated modelling system can be used to enable comprehensive analyses of agri-environmental policies that are targeted at the individual farm level. It will be worth exploring whether the slight deviation (like 1%) of the surrogate model at the farm level can cause crucial divergence at the regional level, where heterogeneous farms interact with each other in both the short and long run. Furthermore, updating and debugging the integrated modelling system could be challenging because three different models (i.e., farm model, surrogate model and ABM) that are potentially operated by different teams are involved.

Finally, future research may move towards more systematic development and integrated application of surrogate models going beyond their stand-alone methodological assessment. An interesting alternative avenue in training surrogate models might be the use of generative adversarial networks (GANs) (Goodfellow et al., 2014). They could learn the criteria for making the outcomes from the original and surrogate model indistinguishable in a data-driven way or could allow us to derive more natural stopping criteria for data generation. The rapid development of machine learning will likely further improve the performance of surrogate models and make the training of NNs a more standard approach.

ACKNOWLEDGEMENTS

This work received funding from the European Union's research and innovation programme under grant agreement No. 817566 – MIND STEP. It is also partially funded by the German Research Foundation under Germany's Excellence Strategy, EXC-2070-390732324-PhenoRob. Our thanks are also due to anonymous reviewers for their constructive comments on an earlier draft. Open Access funding enabled and organized by Projekt DEAL.

DATA AVAILABILITY STATEMENT

The data and code used for this paper can be found in the following Github repository: https://github.com/linmeishang/SurrogateNN.

ORCID

Linmei Shang https://orcid.org/0000-0002-5044-3219
Franziska Appel https://orcid.org/0000-0002-6049-4511

ENDNOTES

[1] See Goodfellow et al. (2016) for further details.

[2] See the Github repository of the 'Keras' library (Chollet, 2015).

[3] In the context of OLS (ordinary least squares), R^2 will always be between 0 and 1. But in the machine learning context, when R^2 is calculated based on a test set not used in estimation, negative values may occur even if the fit criterion in training is least squares.

[4] In practice, this threshold is <0.01 because NNs usually do not predict a strict '0' but rather a very small number like 0.000001.

[5] This is the maximum amount of observations in the original training set.

[6] Because of the poor performance, the evaluations of ResNet with 1000 observations are not shown in the following figures, either.

REFERENCES

Albanese, D., Filosi, M., Visintainer, R., Riccadonna, S., Jurman, G. & Furlanello, C. (2013) Minerva and minepy: a C engine for the MINE suite and its R, python and MATLAB wrappers. *Bioinformatics (Oxford, England)*, 29, 407–408.

Alibabaei, K., Gaspar, P.D. & Lima, T.M. (2021) Modeling soil water content and reference evapotranspiration from climate data using deep learning method. *Applied Sciences*, 11, 5029.

Amouzgar, K. & Strömberg, N. (2017) Radial basis functions as surrogate models with a priori bias in comparison with a posteriori bias. *Structural and Multidisciplinary Optimization*, 55, 1453–1469.

An, L., Grimm, V., Sullivan, A., Turner, B.L., II, Malleson, N., Heppenstall, A. et al. (2021) Challenges, tasks, and opportunities in modeling agent-based complex systems. *Ecological Modelling*, 457, 109685.

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

Appel, F., Ostermeyer-Wiethaup, A. & Balmann, A. (2016) Effects of the German renewable energy act on structural change in agriculture – the case of biogas. *Utilities Policy*, 41, 172–182.

Audsley, E., Pearn, K.R., Harrison, P.A. & Berry, P.M. (2008) The impact of future socio-economic and climate changes on agricultural land use and the wider environment in East Anglia and north West England using a metamodel system. *Climatic Change*, 90, 57–88.

Bengio, Y., Simard, P. & Frasconi, P. (1994) Learning long-term dependencies with gradient descent is difficult. *IEEE Transactions on Neural Networks*, 5, 157–166.

Bradhurst, R.A., Roche, S.E., East, I.J., Kwan, P. & Garner, M.G. (2016) Improving the computational efficiency of an agent-based spatiotemporal model of livestock disease spread and control. *Environmental Modelling & Software*, 77, 1–12.

Britz, W. (2021) Automated calibration of farm-Sale mixed linear programming models using Bi-level programming. *German Journal of Agricultural Economics*, 70, 165–181.

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

Britz, W., Lengers, B., Kuhn, T. & Schäfer, D. (2016) *A highly detailed template model for dynamic optimization of farms – FARMDYN*. Bonn: Institute for Food and Resource Economics, University of Bonn. Available from: https://www.ilr.uni-bonn.de/em/rsrch/farmdyn/farmdyn_docu.pdf [Accessed 03rd April 2022]

Cao, D., Chen, Y., Chen, J., Zhang, H. & Yuan, Z. (2021) An improved algorithm for the maximal information coefficient and its application. *Royal Society Open Science*, 8, 201424.

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

Chen, R., Zhang, W. & Wang, X. (2020) Machine learning in tropical cyclone forecast modeling: a review. *Atmosphere*, 11, 676.

Chen, X., Chen, R., Wan, Q., Xu, R. & Liu, J. (2021) An improved data-free surrogate model for solving partial differential equations using deep neural networks. *Scientific Reports*, 11, 19507.

Chollet, F. (2015) *Keras* (GitHub, 2015). Available from: https://github.com/fchollet/keras [Accessed 03rd April 2022]

Chopra, C., Sinha, S., Jaroli, S., Shukla, A. & Maheshwari, S. (2017) Recurrent Neural Networks with Non-Sequential Data to Predict Hospital Readmission of Diabetic Patients. *Proceedings of the 2017*

International Conference on Computational Biology and Bioinformatics – ICCBB 2017, Newark, NJ, USA, 18/10/2017–20/10/2017.

Debertin, D.L. (2012) *Agricultural production economics*. New York: Macmillan Publishing Company.

Elman, J. (1990) Finding structure in time. *Cognitive Science*, 14, 179–211.

Fallah-Mehdipour, E., Bozorg Haddad, O. & Mariño, M.A. (2013) Prediction and simulation of monthly groundwater levels by genetic programming. *Journal of Hydro-Environment Research*, 7, 253–260.

Fawaz, H.I., Forestier, G., Weber, J., Idoumghar, L. & Muller, P.-A. (2019) Deep learning for time series classification: a review. *Data Mining and Knowledge Discovery*, 33, 917–963.

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

Goodfellow, I., Bengio, Y. & Courville, A. (2016) *Deep learning*. Cambridge, MA: The MIT Press.

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S. et al. (2014) Generative adversarial networks. In: *Advances in neural information processing systems*. Cambridge, MA: MIT Press, pp. 2672–2680.

Graves, A., Fernández, S. & Schmidhuber, J. (2005) Bidirectional LSTM networks for improved phoneme classification and recognition. In: Hutchison, D., Kanade, T., Kittler, J., Kleinberg, J.M., Mattern, F., Mitchell, J.C. et al. (Eds.) *Artificial neural networks: formal models and their applications – ICANN 2005*. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 799–804.

Graves, A., Mohamed, A.-R. & Hinton, G. (2013) Speech recognition with deep recurrent neural networks. Available from: http://arxiv.org/pdf/1303.5778v1 [Accessed 03rd April 2022]

Gruber, A., Yanovski, S. & Ben-Gal, I. (2013) Condition-based maintenance via simulation and a targeted Bayesian network metamodel. *Quality Engineering*, 25, 370–384.

Happe, K., Balmann, A., Kellermann, K. & Sahrbacher, C. (2008) Does structure matter? The impact of switching the agricultural policy regime on farm structures. *Journal of Economic Behavior & Organization*, 67, 431–444.

Happe, K., Kellermann, K. & Balmann, A. (2006) Agent-based analysis of agricultural policies: an illustration of the agricultural policy simulator AgriPoliS, its adaptation and behavior. *Ecology and Society*, 11, 49.

He, K., Zhang, X., Ren, S. & Sun, J. (2016) Deep Residual Learning for Image Recognition. In: *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA*. Piscataway, NJ: IEEE, pp. 770–778.

Heckelei, T. (2013) General methodological issues on farm level modelling. In: Langrell, S. (Ed.) *Farm level modelling of CAP: a methodological overview*. Luxembourg: Publications Office, pp. 29–34.

Heinrichs, J., Jouan, J., Pahmeyer, C. & Britz, W. (2021) Integrated assessment of legume production challenged by European policy interaction: a case-study approach from French and German dairy farms. *Q Open*, 1, qoaa011.

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

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

Hsu, D. (2017) Multi-period Time Series Modeling with Sparsity via Bayesian Variational Inference. Available from: https://arxiv.org/abs/1707.00666v3 [Accessed 03rd April 2022].

Huang, L., Qin, J., Zhou, Y., Zhu, F., Liu, L. & Shao, L. (2020) Normalization techniques in training DNNs: methodology, analysis and application. https://doi.org/10.48550/arXiv.2009.12836 [Accessed 03rd April 2022].

Huber, R., Bakker, M., Balmann, A., Berger, T., Bithell, M., Brown, C. et al. (2018) Representation of decision-making in European agricultural agent-based models. *Agricultural Systems*, 167, 143–160.

Huber, R., Xiong, H., Keller, K. & Finger, R. (2022) Bridging behavioural factors and standard bio-economic modelling in an agent-based modelling framework. *Journal of Agricultural Economics*, 73, 35–63.

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

Jäger, G. (2021) Using neural networks for a universal framework for agent-based models. *Mathematical and Computer Modelling of Dynamical Systems*, 27, 162–178.

Jiang, P., Zhou, Q. & Shao, X. (2020) *Surrogate model-based engineering design and optimization*. Springer Singapore: Singapore.

Kleijnen, J.P.C. (2009) Kriging metamodeling in simulation: a review. *European Journal of Operational Research*, 192, 707–716.

Kremmydas, D., Athanasiadis, I.N. & Rozakis, S. (2018) A review of agent based modeling for agricultural policy evaluation. *Agricultural Systems*, 164, 95–106.

Kuhfuss, L., Préget, R., Thoyer, S. & Hanley, N. (2016) Nudging farmers to enrol land into Agri-environmental schemes: the role of a collective bonus. *European Review of Agricultural Economics*, 43, 609–636.

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

Kuhn, T., Möhring, N., Töpel, A., Jakob, F., Britz, W., Bröring, S. et al. (2022) Using a bio-economic farm model to evaluate the economic potential and pesticide load reduction of the greenRelease technology. *Agricultural Systems*, 201, 103454.

LeCun, Y., Boser, B., Denker, J., Henderson, D., Howard, R., Hubbard, W. et al. (1990) Handwritten digit recognition with a Back-propagation network. In: Touretzky, D.S. (Ed.) *Neural information processing systems. Natural and synthetic conference papers*. Denver: Morgan Kaufmann.

Liong, S.-Y., Khu, S.-T. & Chan, W.-T. (2001) Derivation of pareto front with genetic algorithm and neural network. *Journal of Hydrologic Engineering*, 6, 52–61.

Marhon, S.A., Cameron, C.J.F. & Kremer, S.C. (2013) Recurrent neural networks. In: Bianchini, M., Maggini, M. & Jain, L.C. (Eds.) *Handbook on neural information processing*. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 29–65.

McKay, M.D., Beckman, R.J. & Conover, W.J. (1979) A comparison of three methods for selecting values of input variables in the analysis of output from a computer code. *Technometrics*, 21, 239.

Müller, B., Hoffmann, F., Heckelei, T., Müller, C., Hertel, T.W., Polhill, J.G. et al. (2020) Modelling food security: bridging the gap between the micro and the macro scale. *Global Environmental Change*, 63, 102085.

Murray-Rust, D., Brown, C., van Vliet, J., Alam, S.J., Robinson, D.T., Verburg, P.H. et al. (2014) Combining agent functional types, capitals and services to model land use dynamics. *Environmental Modelling & Software*, 59, 187–201.

Nguyen, T.H., Nong, D. & Paustian, K. (2019) Surrogate-based multi-objective optimization of management options for agricultural landscapes using artificial neural networks'. *Ecological Modelling*, 400, 1–13.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G. et al. (2019) Pytorch: An imperative style, high-performance deep learning library. In: Wallach, H., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E. & Garnett, R. (Eds.) *Advances in neural information processing systems*, Vol. 32. Red Hook, NY: Curran Associates, Inc, pp. 8024–8035.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O. et al. (2011) Scikit-learn: machine learning in python. Available from: http://arxiv.org/pdf/1201.0490v4 [Accessed 03rd April 2022]

Picheny, V. (2015) Multiobjective optimization using gaussian process emulators via stepwise uncertainty reduction. *Statistics and Computing*, 25, 1265–1280.

Poppe, K., Duinen, L. & Koeijer, T. (2021) Reduction of greenhouse gases from peat soils in Dutch agriculture. *EuroChoices*, 20, 38–45.

Rahmani, F., Lawson, K., Ouyang, W., Appling, A., Oliver, S. & Shen, C. (2021) Exploring the exceptional performance of a deep learning stream temperature model and the value of streamflow data. *Environmental Research Letters*, 16, 24025.

Rasch, S., Heckelei, T., Storm, H., Oomen, R. & Naumann, C. (2017) Multi-scale resilience of a communal rangeland system in South Africa. *Ecological Economics*, 131, 129–138.

Ratto, M., Castelletti, A. & Pagano, A. (2012) Emulation techniques for the reduction and sensitivity analysis of complex environmental models. *Environmental Modelling & Software*, 34, 1–4.

Razavi, S. (2021) Deep learning, explained: fundamentals, Explainability, and Bridgeability to process-based modelling. *Environmental Modelling & Software*, 144, 105159.

Razavi, S., Tolson, B.A. & Burn, D.H. (2012) Review of surrogate modeling in water resources. *Water Resources Research*, 48, 559.

Reshef, D.N., Reshef, Y.A., Finucane, H.K., Grossman, S.R., McVean, G., Turnbaugh, P.J. et al. (2011) Detecting novel associations in large data sets. *Science (New York, N.Y.)*, 334, 1518–1524.

Richardson, J.W., Hennessy, T. & O'Donoghue, C. (2014) Farm Level Models. In: O'Donoghue, C. (Ed.) *Handbook of microsimulation modelling*. Bingley: Emerald Group Publishing Limited, pp. 505–534.

Roman, N.D., Bre, F., Fachinotti, V.D. & Lamberts, R. (2020) Application and characterization of metamodels based on artificial neural networks for building performance simulation: a systematic review. *Energy and Buildings*, 217, 109972.

Rumelhart, D.E., Hinton, G.E. & Williams, R.J. (1986) Learning representations by back-propagating errors. *Nature*, 323, 533–536.

Seidel, C. & Britz, W. (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. & Rasch, S. (2021) Adoption and diffusion of digital farming technologies – integrating farm-level evidence and system interaction. *Agricultural Systems*, 190, 103074.

Storm, H., Baylis, K. & Heckelei, T. (2020) Machine learning in agricultural and applied economics. *European Review of Agricultural Economics*, 47, 849–892.

Šumrada, T., Japelj, A., Verbič, M. & Erjavec, E. (2022) Farmers' preferences for result-based schemes for grassland conservation in Slovenia. *Journal for Nature Conservation*, 66, 126143.

Sun, G. & Wang, S. (2019) A review of the artificial neural network surrogate modeling in aerodynamic design. *Proceedings of the Institution of Mechanical Engineers, Part G: Journal of Aerospace Engineering*, 0, 1–10.

Sun, J., Di, L., Sun, Z., Shen, Y. & Lai, Z. (2019) County-level soybean yield prediction using deep CNN-LSTM model. *Sensors (Basel, Switzerland)*, 19, 4363.

Sun, Z., Lorscheid, I., Millington, J.D., Lauf, S., Magliocca, N.R., Groeneveld, J. et al. (2016) Simple or complicated agent-based models? A complicated issue. *Environmental Modelling & Software*, 86, 56–67.

Tian, H., Wang, P., Tansey, K., Zhang, J., Zhang, S. & Li, H. (2021) An LSTM neural network for improving wheat yield estimates by integrating remote sensing data and meteorological data in the Guanzhong plain, Pr China. *Agricultural and Forest Meteorology*, 310, 108629.

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*. Available from: https://doi.org/10.1007/s10614-022-10276-0

Tyan, M. & Lee, J.-W. (2019) Efficient multi-response adaptive sampling algorithm for construction of variable-fidelity aerodynamic tables. *Chinese Journal of Aeronautics*, 32, 547–558.

Weber, T., Corotan, A., Hutchinson, B., Kravitz, B. & Link, R. (2019) Technical Note: Deep Learning for Creating Surrogate Models of Precipitation in Earth System Models.

Weersink, A., Jeffrey, S. & Pannell, D. (2002) Farm-level modeling for bigger issues. *Review of Agricultural Economics*, 24, 123–140.

Werbos, P.J. (1988) Generalization of backpropagation with application to a recurrent gas market model. *Neural Networks*, 1, 339–356.

Xiang, H., Li, Y., Liao, H. & Li, C. (2017) An adaptive surrogate model based on support vector regression and its application to the optimization of railway wind barriers. *Structural and Multidisciplinary Optimization*, 55, 701–713.

SUPPORTING INFORMATION

Additional supporting information can be found online in the Supporting Information section at the end of this article.

How to cite this article: Shang, L., Wang, J., Schäfer, D., Heckelei, T., Gall, J., Appel, F. et al. (2023) Surrogate modelling of a detailed farm-level model using deep learning. *Journal of Agricultural Economics*, 00, 1–26. Available from: https://doi.org/10.1111/1477-9552.12543