Journal of Productivity Analysis (2026) 65:13
https://doi.org/10.1007/s11123-026-00797-3

Shape-aware deep learning for models of production

Zheng Wei[1] · Huiyan Sang[2] · Artem Prokhorov[3,4,5] · Yu Ma[1]

Received: 27 October 2025 / Accepted: 2 February 2026

Abstract
The stochastic frontier model (SFM) is widely employed in the analysis of productivity and efficiency, yet strict parametric forms, such as the Cobb-Douglas and Translog functions, are often assumed for modeling production, leading to potential misspecification issues. While semi- and nonparametric SFMs offer greater flexibility, they face challenges in imposing monotonicity and concavity to maintain their desirable economic interpretation. We develop a framework which enforces the shape restrictions within deep neural networks (DNNs). The stochastic frontier model we develop (DNN-SFM) leverages the flexibility and predictive power of DNNs while preserving key properties of a production function, such as free disposability and diminishing marginal product. Additionally, we demonstrate how to use Shapley values to measure and interpret global and local effects of individual inputs on the production frontier in cases when model parameters to not admit a simple interpretation. The performance of the proposed method is assessed using simulations while a real-world application to rice production in the Philippines illustrates empirical relevance of the proposed method.

Keywords Deep neural networks · Stochastic frontier models · Shapley values

1 Introduction

The canonical parametric stochastic frontier model (SFM) for a cross-section of n decision-making units can be expressed as follows (Aigner et al. 1977; Meeusen and van Den Broeck 1977):

$$Y_i = f(\boldsymbol{X}_i;\beta) + V_i - U_i, \qquad i = 1,\ldots,n, \tag{1}$$

where Y_i is log-output, $f(\cdot\,;\boldsymbol{\beta})$ is a production function of log-inputs $\boldsymbol{X}_i \in \mathbb{R}^K$ parameterized by $\boldsymbol{\beta}$, V_i is a random error and $U_i \geq 0$ is technical inefficiency. Production functions satisfy several well-established shape constrains, namely, the free disposability (monotonicity) and diminishing marginal returns (concavity) (see, e.g., Prokhorov 2024, Chapter 2).

A parametric framework often pre-specifies an explicit functional form of the boundary of the feasible production set. Commonly used functional forms include the Cobb-Douglas and Translog functions (see, e.g., Zellner et al. 1966). While these production functions are widely used, the rigid parametric assumptions could result in model misspecifications. Prior research (see, e.g., Ferrara and Vidoli 2017; Giannakas et al. 2003; Parmeter et al. 2017; Wei et al. 2024) has highlighted that many conventional frontier specifications are overly restrictive and fail to capture the complexity of real-world production frontiers. These limitations can introduce substantial bias, leading to inaccurate interpretations of input-output relationships and misguided policy recommendations.

Researchers have developed numerous nonparametric and semiparametric versions of the SFM to relax the parametric functional form specifications, including a growing number of machine learning approaches. For example,

✉ Zheng Wei
zheng.wei@tamucc.edu

[1] Department of Mathematics and Statistics, Texas A&M University - Corpus Christi, Corpus Christi, US

[2] Department of Statistics, Texas A&M University, College Station, US

[3] Discipline of Business Analytics, University of Sydney, Sydney, Australia

[4] Center for Big Data in Economics and Finance (CEBDA), HSE University, Moscow, Russia

[5] Centre interuniversitaire de recherche en économie quantitative (CIREQ), Université de Montréal, Montréal, Canada

Published online: 23 February 2026

Fan et al. (1996) introduced a two-step pseudo-likelihood procedure using kernel regressions for frontier estimation. Kumbhakar et al. (2007) proposed a local maximum likelihood approach. Ferrara and Vidoli (2017) introduced a semiparametric approach using generalized additive models with spline-based smooth functions of each input variable (GAM-SFM). Tran et al. (2023) developed a two-step semiparametric estimation procedure for a spatial autoregressive SFM, as an extension of the parametric model with endogeneity of Kutlu et al. (2020). Tsionas and Mamatzakis (2019) proposed using an artificial neural network to approximate the inefficiency effects in SFM. Tsionas (2020) introduced a nonparametric functional form into a quantile SFM. More recently, Kutlu and Mao (2024) applied neural networks in the context of nonparametric SFM. The list of such nonparametric approaches can be extended and is invariably incomplete.

Guaranteeing that any such method satisfies the shape constraints is a difficult task. Even when such a guarantee is provided, we face other limitations of nonparametric approaches. For example, a key limitation of GAM-SFM, which ensures monotonicity, is its inability to efficiently capture high-dimensional input interactions, as the dimensionality of the tensor basis increases exponentially with the number of inputs. Similarly, a two-stage approach by Kuosmanen and Kortelainen (2012) known as the Stochastic Non-smooth Envelopment of Data (StoNED), which imposes the required constraints through the convex nonparametric least squares of Hanson and Pledger (1976), is not designed to effectively handle high-dimensional inputs and input interactions, besides being non-smooth.

Parmeter and Racine (2012) proposed a smooth constrained kernel-based method that adjusts conventional regression smoothers using observation specific weights designed to make the function and its derivatives stay within prespecified bounds, thus enforcing axioms of production. The use of multivariate kernels implicitly accommodates nonlinear and higher-order interactions. However, bandwidth selection in multivariate kernel regressions is challenging in moderate to high dimensions. As they mention, nothing prevents their approach from being used for other nonparametric estimators, including artificial neural networks, but they do not pursue this.

In a Bayesian framework, Tsionas (2022a) proposed a regression tree approach for efficiency estimation using decision trees. This imposed the monotone and concave constraints but only within each subregion of the input space corresponding to each decision tree leaf node. Wei et al. (2024) used monotone Bayesian additive regression trees to estimate SFMs. Tsionas et al. (2023) used neural networks to approximate the deterministic inefficiency and to model the noise as a smooth mixture of normal distributions. The shape constraints of these models, especially the concavity constraint, are imposed through rejection sampling in MCMC, i.e., by checking whether each generated sample satisfies the desired shape constraints on a selected grid of input points. Such algorithms are computationally very expensive or even infeasible for high-dimensional input spaces with possible interactions.

Recently, deep learning has demonstrated remarkable breakthroughs in prediction problems across various fields, ranging from fraud detection to medical diagnostics to financial forecasting (see, e.g., Bishop and Bishop 2023; LeCun et al. 2015). The popular neural network architectures driving modern applications in this area include deep neural networks (DNNs), convolutional neural networks (CNNs), recurrent neural networks (RNNs), autoencoders, generative adversarial networks (GANs), transformers, and large language models (LLMs). Among these, feed-forward DNNs serve as the foundation for all deep learning models and are credited for their exceptional performance.

DNN architectures have been used in models of production before (see, e.g., Kutlu and Mao 2024; Tsionas et al. 2023; Tsionas 2022b; Wang 1996, 2003). However, no paper known to us has developed a general class of activation functions that respects axioms of production by enforcing monotonicity and concavity of the resulting production function in deep learning settings. One of the key contributions of this paper is to show that for a DNN to satisfy the shape constraints by construction it has to use a composition of non-standard shape-constrained activation functions with nonnegative parameters, something we call *shape-aware* learning.

The proposed DNN-SFMs have several advantages over existing SFMs, in addition to satisfying both the free disposability and diminishing marginal returns properties. First, the DNN-SFMs offer greater flexibility in modeling complex nonlinear relationships in SFMs, allowing for a more accurate representation of production functions without the need for explicit specifications of functional forms. Second, the DNN-SFMs handle large and complex input data due to automatic differentiation used in gradient-based backpropagation on graphics processing units (GPUs). Finally, they allow us to integrate explainability and visualization by using Shapley values to understand how the various factors affect production. While we use Shapley values to interpret the fitted DNN-SFM, it is important to recognize that Shapley values are not structural objects. They capture how changes in an input are associated with changes in the model's predicted output, averaging over the joint distribution of other inputs, but they do not generally coincide with causal or structural partial effects, such as marginal products. Such an interpretation would require substantial additional assumptions and is left for future work.

Our findings show that DNN-SFM represents a substantive methodological advance over existing approaches. In our Monte Carlo simulations, the shape-aware DNN-SFM consistently outperforms both classical parametric models and popular semi- and nonparametric alternatives in terms of accuracy of frontier recovery, inefficiency estimation, and robustness to irrelevant regressors. Unlike unconstrained counterparts, the proposed shape-aware architecture enforces globally concave production frontiers that are closer to the data-generating process. In the empirical application to Philippine rice production, the DNN-SFM avoids the "wrong skewness" problem that renders the conventional linear SFM unusable, produces economically sensible associations, and yields markedly different efficiency rankings with meaningful heterogeneity across farms. Taken together, these results suggest that DNN-SFM is an attractive framework that combines the adaptivity and predictive power of DNNs with credible production and efficiency analysis.

The rest of the paper is organized as follows. Section 2 introduces the DNN-SFM. In Section 2.3, we provide sufficient conditions for neural networks to satisfy the monotonicity and concavity constraints. Section 3 discusses the estimation of DNN-SFMs and technical efficiency scores. Section 3.3 addresses the question of how to use Shapley values in DNN-SFM. Section 4 includes simulations with a comparison of several competing methods. Section 5 conducts a real data analysis, which highlights the advantage of DNN-SFM. Finally, Section 6 gives our concluding remarks. The codes and data used for simulations and application are available in Python on the authors' web pages and at a GitHub repository.[1]

2 DNN-SFM

2.1 Model setup

The DNN model can be viewed as a highly flexible nonparametric extension of traditional regression models to non-linear functions via the incorporation of multiple hidden layers and activation functions. By the universal approximation theorem (see, e.g., Cybenko 1989; Hornik et al. 1989), a neural network with one hidden layer can approximate any continuous function $f(\boldsymbol{x})$ to any desired degree of accuracy. However, single-layer networks can approximate complex functions only in principle. Multi-layer networks offer greater computational efficiency and better generalization while achieving the same or better function approximation with a more parsimonious network architecture, especially when modeling highly nonlinear or complex functions (Hornik 1991; Poggio et al. 2017).

The DNN-SFM we propose is semiparametric in the sense that it combines a DNN of a production function with classical distributional assumptions on the composed error term. The model can be written as follows:

$$Y_i = \text{DNN}(\boldsymbol{X}_i; \boldsymbol{w}) + V_i - U_i \quad i = 1, \ldots, n, \tag{2}$$

where DNN($\boldsymbol{X}_i$; $\boldsymbol{w}$) denotes a DNN with L hidden layers, the matrix $\boldsymbol{w} = \{(W^{(\ell)}, \boldsymbol{w}_0^{(\ell)} : \ell = 1, \ldots, L)\}$ combines all DNN parameters, consisting of weights W^{ℓ} and biases $\boldsymbol{w}_0^{(\ell)}$, associated with the ℓ-th hidden layer, l = 1, …, L. Following the standard assumptions in the SFM literature, we assume that the symmetric part of the composed error is normal, $V_i \sim \text{N}(0, \sigma_v^2)$, inefficiency is half-normal, $U_i \sim \text{HN}(0, \sigma_u^2)$, and that V_i and U_i are independent.

To make things more explicit, let hidden layer l have $K^{(\ell)}$ nodes, or neurons, and let $\boldsymbol{h}^{(\ell)} = (h_1^{(\ell)}, \ldots, h_{K^{(\ell)}}^{(\ell)})^{\top}$ denote the node signals in layer l. Then, $W^{(\ell)}$ is a weight matrix with dimensions $K^{(\ell)} \times K^{(\ell-1)}$ and $\boldsymbol{w}_0^{(\ell)}$ is a bias vector of dimension $K^{(\ell)}$. Their purpose is to transform signals from hidden layer $l-1$ to hidden layer l as follows:

$$\underbrace{\begin{pmatrix} h_1^{(\ell)} \\ h_2^{(\ell)} \\ \vdots \\ h_{K^{(\ell)}}^{(\ell)} \end{pmatrix}}_{\boldsymbol{h}^{(\ell)}} = \begin{pmatrix} \sigma^{(\ell)}(o_1^{(\ell)}) \\ \sigma^{(\ell)}(o_2^{(\ell)}) \\ \vdots \\ \sigma^{(\ell)}(o_{K^{(\ell)}}^{(\ell)}) \end{pmatrix} \equiv \sigma^{(\ell)}(\boldsymbol{o}^{(\ell)}), \tag{3}$$

where

$$\underbrace{\begin{pmatrix} o_1^{(\ell)} \\ o_2^{(\ell)} \\ \vdots \\ o_{K^{(\ell)}}^{(\ell)} \end{pmatrix}}_{\boldsymbol{o}^{(\ell)}} = \underbrace{\begin{pmatrix} w_{1,1}^{(\ell)} & w_{1,2}^{(\ell)} & \cdots & w_{1,K^{(\ell-1)}}^{(\ell)} \\ w_{2,1}^{(\ell)} & w_{2,2}^{(\ell)} & \cdots & w_{2,K^{(\ell-1)}}^{(\ell)} \\ \vdots & \vdots & & \vdots \\ w_{K^{(\ell)},1}^{(\ell)} & w_{K^{(\ell)},2}^{(\ell)} & \cdots & w_{K^{(\ell)},K^{(\ell-1)}}^{(\ell)} \end{pmatrix}}_{W^{(\ell)}} \underbrace{\begin{pmatrix} h_1^{(\ell-1)} \\ h_2^{(\ell-1)} \\ \vdots \\ h_{K^{(\ell-1)}}^{(\ell-1)} \end{pmatrix}}_{\boldsymbol{h}^{(\ell-1)}} + \underbrace{\begin{pmatrix} w_{01}^{(\ell)} \\ w_{02}^{(\ell)} \\ \vdots \\ w_{0K^{(\ell)}}^{(\ell)} \end{pmatrix}}_{\boldsymbol{w}_0^{(\ell)}}$$

and $\sigma^{(\ell)}$ is a pre-specified activation function which operates elementwise on its vector argument.

It is clear that the signal of layer zero is the K-vector of inputs, $\boldsymbol{h}^{(0)} = \boldsymbol{X} = (X_1, \ldots, X_K)^{\top}$, and that the DNN-SFM can be expressed in matrix form as follows:

$$Y_i = \boldsymbol{h}^{(L)} + V_i - U_i \tag{4}$$

where $\boldsymbol{h}^{(L)}$ is defined recursively as follows:

$$\boldsymbol{h}^{(\ell)} = \sigma^{(\ell)}\left(W^{(\ell)}\boldsymbol{h}^{(\ell-1)} + \boldsymbol{w}_0^{\ell}\right), \qquad \text{for } \ell = 1, \ldots, L.$$

[1] The repository can be found at https://github.com/ZWeiSTAT/shape-constrained_neural_networks.

A neural network is deep if $L > 1$ and it is feed-forward if there are no loops between its nodes. Figure 1 illustrates a DNN, for which $K = 2$, $L = 3$ and $K^{(1)} = K^{(2)} = K^{(3)} = 5$. Given a loss function, the DNN parameters $\boldsymbol{w}$ are estimated, or trained, by backpropagation, which is a way of calculating the gradient of the loss function with respect to each parameter using the chain rule in a backward pass from the output layer to the input layer.

A natural econometric question is whether the DNN parameters $\boldsymbol{w}$ are uniquely identified. Unfortunately, the DNN uniform approximation theorems are existence results, not uniqueness results. It is well-known that due to permutation symmetries, scaling symmetries, and overparameterization, DNN parameter configurations are generally not unique. For certain shallow networks (one hidden layer) with specific activations and assumptions, one can get identifiability up to permutations. For deep networks, different configurations of weights and biases can produce the same function (exactly) or be observationally equivalent on any given dataset (approximately) (see, e.g., DeVore et al. 2021). In applications such as ours, it is common to interpret the estimated function (or derived quantities like marginal effects, SHAP or efficiency scores) rather than the raw DNN weights.

2.2 Limitations of popular activation functions

The use of an activation function in DNN is inspired by biological neurons in the nervous system, and its choice plays a crucial role in the function approximation performance of DNN in practice. Activation functions introduce nonlinearity into the model, enabling it to learn and approximate complex patterns in the data. We consider two commonly used activation functions:

(a) Rectified Linear Unit (ReLU):

$$\sigma(t) = \max(0, t) = \begin{cases} t & t \geq 0, \\ 0 & t < 0. \end{cases} \tag{5}$$

This is one of the most widely used activation functions in machine learning due to its computational efficiency and ability to mitigate the issue of vanishing gradients (see, e.g., Maas et al. 2013; Nair and Hinton 2010). Vanishing gradients are a training problem in DNNs where gradients, which are used to update the DNN parameters, become progressively smaller during backpropagation, due to repeatedly multiplying derivatives of the activation function according to the chain rule. For this reason, activation functions with derivatives less than one such as *sigmoid* and *tanh* cause the early layers of the DNN to learn very slowly or not at all.

(b) Exponential Linear Unit (ELU):

$$\sigma(t) = \begin{cases} t & t > 0, \\ \alpha(e^t - 1) & t \leq 0, \end{cases} \tag{6}$$

where $\alpha > 0$ is a hyperparameter to be tuned. This function was proposed for reducing bias shifts and improving algorithm convergence due to smoother weight updates during backpropagation (Clevert et al. 2016).

There are many other activation functions in the DNN literature (see,e.g., Goodfellow et al. 2016; He et al. 2015; LeCun et al. 1989; Nair and Hinton 2010; Ramachandran et al. 2017). Unfortunately, none of the ones known to us, including ReLU and ELU, necessarily produce DNNs that respect the shape constraints required for a proper production function. We illustrate this issue using a simulation example.

Example 1 (DNN-SFM with ReLU and ELU). In this example, we simulate data with sample size $n = 200$ from the Cobbs-Douglas function with one input:

$$Y_i = \beta_0 X_i^{\beta_1} \exp(V_i - U_i), \qquad i = 1, \ldots, n, \tag{7}$$

where $\beta_0 = 3$, $\beta_1 = 0.2$, $V_i \sim \mathrm{N}(0, \sigma_v^2)$ and $U_i \sim \mathrm{HN}(0, \sigma_u^2)$ with $\sigma_v = 0.2$ and $\sigma_u = 0.1$, and X_i is generated as Unif(0, 5).

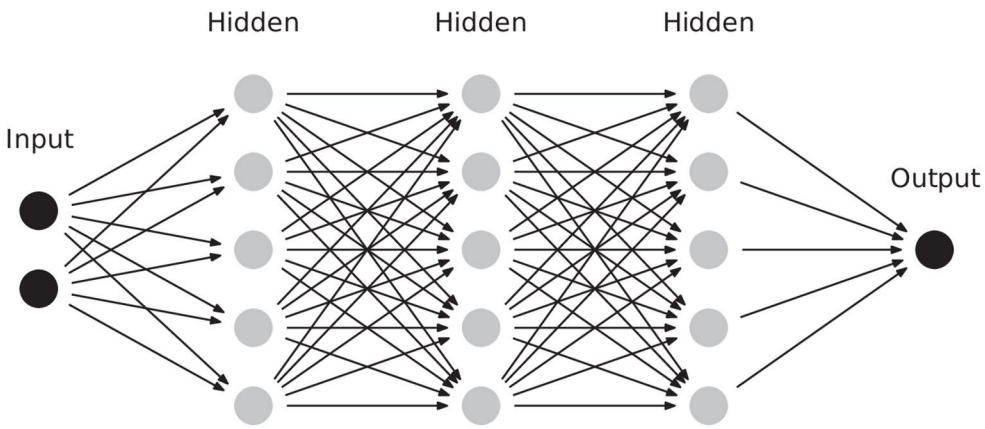

Fig. 1 A feed-forward fully connected deep neural network with one input layer (with two input variables), three hidden layers (with five neurons each), and one output layer

For estimation we use a DNN with $L = 2$ trained using the ReLU and ELU activation functions. Figure 2 shows the simulated data (gray dots), true function (solid curve) and fitted curve (dashed curve). As can be seen from the plots, the fitted DNNs are not globally monotone or concave in both cases.

2.3 Shape-aware DNN-SFM

Production functions are non-negatively weakly monotone increasing and quasi-concave, which means the technology or production possibility set is convex and obeys the axioms of production, such as free disposability and diminishing marginal returns (see, e.g., Kumbhakar and Lovell 2003, Section 2.2). Monotonicity means that the output cannot decrease with an increase in inputs. Concavity reflects diminishing marginal product, where the incremental contribution to the output of each additional unit of input decreases as the input increases. We seek to characterize DNNs that respect these shape constraints by construction.

The following theorem provides sufficient conditions for a DNN to satisfy the shape constraints of a production function.

Theorem 1 A neural network DNN($\boldsymbol{X}$; $\boldsymbol{w}$) is a production function satisfying all the axioms of production if the following conditions are satisfied for $\ell = 1, \ldots, L$:

1. The activation functions $\sigma^{(\ell)}(\cdot)$ are concave and non-decreasing and $\sigma^{(\ell)}(0) = 0$;
2. The weights and biases of all hidden layers, $W^{(\ell)}$ and $\boldsymbol{w}_0^{\ell}$, are non-negative.

Proof. From the definition in Eq. (4), DNN($\boldsymbol{X}$; $\boldsymbol{w}$) can be expressed as a composition of activation functions $\sigma^{(\ell)}$ and affine transformations using $W^{(\ell)}$, i.e.,

$$\mathrm{DNN}(\boldsymbol{X};\boldsymbol{w}) = (\sigma^{(L)} \circ W^{(L)} \circ \cdots \sigma^{(1)} \circ W^{(1)})(\boldsymbol{X}). \tag{8}$$

Since the weights for each layer are non-negative, affine transformations of $\boldsymbol{X}$ are monotone non-decreasing in each component X_k, for $k = 1, \ldots, K$. Compositions of monotone non-decreasing functions are monotone non-decreasing. Thus, the monotonicity in activation functions $\sigma^{(\ell)}$ and affine transformations with non-negative weights provide a monotone non-decreasing DNN in each input variable X_k, $k = 1, \ldots, K$.

To prove concavity, we note that a composition of a concave function with an affine mapping is concave (see Section 3.2.2 in Boyd and Vandenberghe 2004). Thus, if the activation function is concave, the compositions $\sigma^{(\ell)} \circ W^{(\ell)}$, $\ell = 1, \ldots, L$, are concave functions. Furthermore, a composition of a non-decreasing concave function and a concave function is a concave function. This means that DNN($\boldsymbol{X}$; $\boldsymbol{w}$) is a concave function in $\boldsymbol{X} \in \mathbb{R}^K$.

The condition $\sigma^{(\ell)}(0) = 0$ ensures that with zero weights and biases, DNN($\boldsymbol{0}$; $\boldsymbol{0}$) = 0, which allows for the possibility of inaction. Finally, monotone non-decreasing means non-negativity of DNN($\boldsymbol{X}$; $\boldsymbol{w}$). 7D2□

Remark. Since affine mappings are concave and therefore quasi-concave, and a composition of a quasi-concave function with a monotone non-decreasing function is quasi-concave (see Section 3.4 in Boyd and Vandenberghe 2004), this provides a sufficient condition for DNN($\boldsymbol{X}$; $\boldsymbol{w}$) to be quasi-concave. Specifically, the activation functions $\sigma^{(\ell)}$ must be monotone non-decreasing in each layer $\ell = 1, \ldots, L$.

We emphasize that Theorem 1 provides sufficient, but not necessary, conditions for enforcing shape constraints in DNNs. Consequently, we cannot claim that the proposed shape-aware DNN framework is the tightest or most general representation among shape-constrained function approximators. Establishing tightness and conducting a formal expressive-power comparison across shape-constrained function approximators would require substantially broader theoretical development and is left for future research.

Given the result of the Theorem, we propose two new activation functions for shape-aware DNN-SFMs, namely, *Concave* ReLU (CReLU) and *Concave* ELU (CELU):

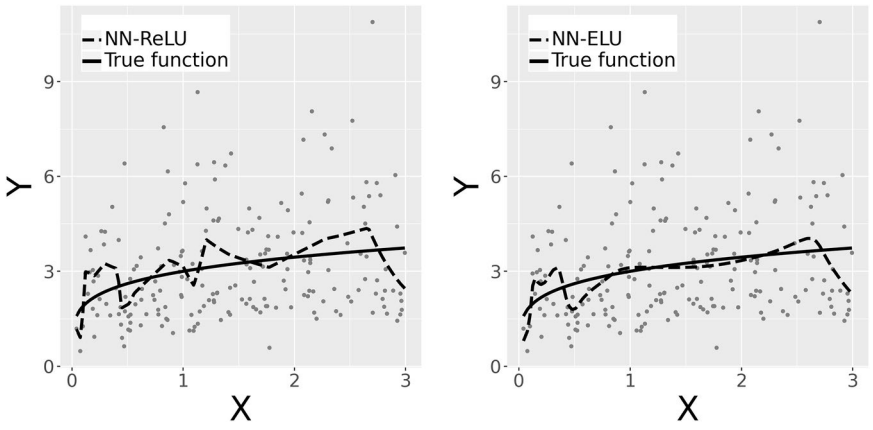

Fig. 2 Example 1 data (gray dots), true function (solid curve) and fitted curve (dashed curve), using the ReLU activation function (left) and ELU activation function (right)

$$[\text{CReLU}] \quad \sigma(t) = \begin{cases} \alpha t, & t \geq 0, \text{with } 0 < \alpha < 1 \\ t, & t < 0. \end{cases} \tag{9}$$

$$[\text{CELU}] \quad \sigma(t) = \begin{cases} -\alpha(e^{-t} - 1), & t \geq 0. \\ t, & t < 0, \end{cases} \tag{10}$$

where, as before, $\alpha > 0$ is a tuning parameter. Example 2 illustrates the appealing features of these activation functions.

Example 2 (Shape-aware DNN-SFM using CReLU and CELU). We return to the data-generating process (DGP) of Example 1 but now use a shape-constrained DNN with new activation functions for training. Figure 3 shows the simulated data (gray dots), true function (solid curve), and fitted DNN-SFM (dashed curve). The figure shows that the fitted shape-aware DNN-SFM is smooth, globally concave and closer to the true production function.

It is natural to ask whether the new activation functions invalidate the universal approximation properties shown to hold for classical activation functions (see, e.g., Cybenko 1989) and whether they suffer from the vanishing gradient problem.

To address the first concern, we note that Yarotsky (2017) established the universal approximation property for neural networks with piecewise linear activation functions, while Sonoda and Murata (2017) extended this result to a broader class of unbounded, piecewise smooth activation functions. The proposed CReLU and CELU functions can be viewed as 180° rotations of the standard ReLU and ELU activations around the origin, which are piecewise linear and piecewise smooth, respectively. Therefore, DNNs employing CReLU and CELU retain the universal approximation property.

To see that CReLU and CELU do not suffer from the vanishing gradient problem the way the *sigmoid* or *tanh* functions do, it is enough to note that CReLU and CELU inherit the functional form of the gradient from ReLU and ELU, which are known to mitigate this problem.

It is worth repeating that CReLU and CELU are not the only admissible concave activations. What matters is not the exact functional form, but whether activation functions satisfy a small set of structural properties listed in Theorem 1 (that is, concave, non-decreasing, zero at the origin), plus are well-behaved for optimization. For example, the following activation functions also satisfy Theorem 1:

$$[\textit{Concave Power Activation}] \quad \sigma(t) = \begin{cases} t, & t \leq 0, \\ t^{\alpha}, & t > 0, \end{cases} \qquad 0 < \alpha < 1.$$

$$[\textit{Log} - \textit{Saturating Activation}] \quad \sigma(t) = \begin{cases} t, & t \leq 0, \\ \log(1+t), & t > 0. \end{cases}$$

$$[\textit{Concave Softplus Activation}] \quad \sigma(t) = \begin{cases} t, & t \leq 0, \\ \alpha \log\left(1 + e^{t/\alpha}\right), & t > 0, \end{cases} \qquad 0 < \alpha < 1.$$

$$[\textit{Piecewise Linear Concave Activation}] \quad \sigma(t) = \begin{cases} t, & t \leq 0, \\ \beta_1 t, & 0 < t \leq c, \\ \beta_2 t + d, & t > c, \end{cases} \qquad \text{with} 1 > \beta_1 > \beta_2 > 0,$$

where $d = (\beta_1 - \beta_2)c$ ensures continuity.

$$[\textit{Negative} - \textit{Exponential Saturation Activation}] \quad \sigma(t) = \begin{cases} t, & t \leq 0, \\ A\left(1 - e^{-Bt}\right), & t > 0, \end{cases} \qquad A, B > 0.$$

$$[\textit{Concave Arctangent Activation}] \quad \sigma(t) = \begin{cases} t, & t \leq 0, \\ \arctan(t), & t > 0. \end{cases}$$

The ultimate choice of an activation function may depend on factors such as numerical stability, smoothness of the output, strength of the marginal effect, and empirical performance on the dataset. For example, CELU offers smooth differentiability that aids gradient-based optimization, while CReLU may be preferred for capturing sharper changes in the production frontier. Concave Softplus Activation is a smooth analogue of CReLU. Concave Power Activation

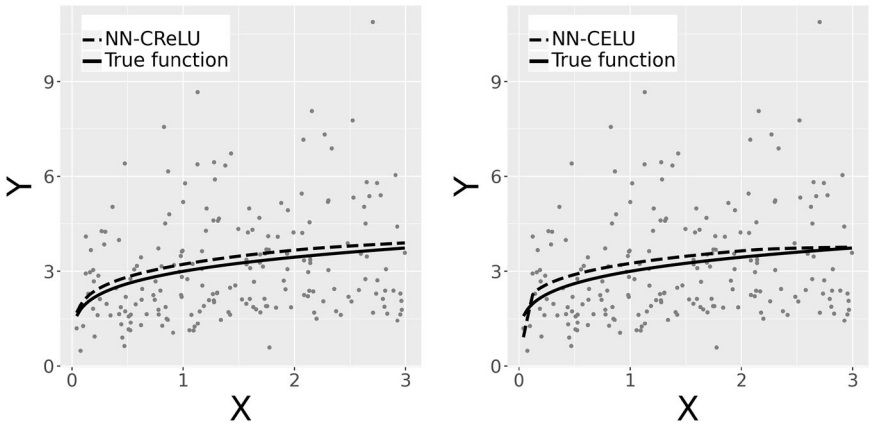

Fig. 3 Example 2 data (gray dots), true function (solid curve) and fitted curve (dashed curve), using the CReLU activation function (left) and CElu activation function (right)

flattens quickly as inputs increase, showing a strong diminishing marginal effect, but its derivative explodes near zero when α is small, which may require careful tuning. Log-Saturating Activation can be restrictive if the frontier is only mildly concave because, due to the saturation effect, large increases in inputs yield small output gains. A piecewise linear concave spline requires choosing knot locations. Negative-Exponential Saturation Activation is bounded which may be undesirable in some applications.

3 Inference and interpretation

3.1 Estimation

In this section, we consider the estimation of the shape-constrained DNN-SFMs. Let $\boldsymbol{\theta}$ denote the unknown model parameters consisting of $\boldsymbol{w} = \{(W^{(\ell)}, \boldsymbol{w}_0^{(\ell)} : \ell = 1, \ldots, L)\}$ in DNN($\boldsymbol{X}_i$; $\boldsymbol{w}$) and the distributional parameters σ_v^2 and σ_u^2, as defined in Section 2.1. Define the composed error term $\mathcal{E}_i = V_i - U_i = Y_i - \text{DNN}(\boldsymbol{X}_i; \boldsymbol{w})$. Then, the density of $\mathcal{E}_i$ can be written as follows

$$\begin{aligned} f_{\mathcal{E}}(\varepsilon_i) &= \int_0^\infty f_v(\varepsilon_i + u) f_u(u) du \\ &= \frac{2}{\sqrt{\sigma_v^2+\sigma_u^2}} \phi\left(\frac{\varepsilon_i}{\sqrt{\sigma_v^2+\sigma_u^2}}\right) \Phi\left(-\frac{\sigma_u \varepsilon_i}{\sigma_v\sqrt{\sigma_v^2+\sigma_u^2}}\right), \end{aligned} \tag{11}$$

where ϕ and Φ are the standard normal density and cumulative distribution function, respectively. The usual estimation of the model parameters $\boldsymbol{\theta} = (\boldsymbol{w}, \sigma_v^2, \sigma_u^2)$ proceeds by minimizing the loss function $L(\cdot)$, based on the negative log-likelihood:

$$L(\boldsymbol{\theta}) = \sum_{i=1}^{n} \{-\ell_i(\boldsymbol{\theta}|\boldsymbol{x}_i, y_i)\}, \tag{12}$$

where

$$\begin{aligned} \ell_i(\boldsymbol{\theta}|\boldsymbol{x}_i, y_i) &= \log f_{\mathcal{E}}(\varepsilon_i) = -\tfrac{1}{2}\log\left(\tfrac{\pi}{2}\right) - \tfrac{1}{2}\log\left(\sqrt{\sigma_v^2+\sigma_u^2}\right) \\ &\quad + \log\Phi\left(-\frac{\sigma_u\varepsilon_i}{\sigma_v\sqrt{\sigma_v^2+\sigma_u^2}}\right) - \frac{\varepsilon_i^2}{2(\sigma_v^2+\sigma_u^2)}. \end{aligned} \tag{13}$$

In the context of DNN, the optimization of $L(\boldsymbol{\theta})$ is performed using the mini-batch stochastic gradient descent (SGD) method, i.e., at the t-th iteration, $\boldsymbol{\theta}_t$ is updated using the following formula:

$$\boldsymbol{\theta}_{t+1} \leftarrow \boldsymbol{\theta}_t - a\frac{1}{|B_t|} \sum_{\boldsymbol{x}_i, y_i \in B_t} \nabla_{\boldsymbol{\theta}} L(\boldsymbol{\theta}), \tag{14}$$

where a is a learning rate and B_t is a batch of size $|B_t|$ from the sample. We have implemented the SGD using autodifferentiation and backpropagation (see, e.g., Rumelhart et al. 1986). It is done via the `Pytorch` library (Paszke et al. 2019), in `Python 3`(Van Rossum and Drake 2009). An advantage of using the new class of activation functions is that we still deal with an unconstrained loss minimization problem and the shape constraints are implicit in the loss, so no additional computational tools from constrained optimization are needed.

Once the model parameters are estimated, technical inefficiency scores (TE) can be computed using the standard analytical form:

$$\text{TE}_i = E(\exp(-U_i)|\varepsilon_i) = \frac{\Phi(\mu_{i*}/\sigma_* - \sigma_*)}{\Phi(\mu_{i*}/\sigma_*)} \exp\left(\frac{1}{2}\sigma_*^2 - \mu_{i*}\right), \tag{15}$$

where $\mu_{i*} = \varepsilon_i \frac{\sigma_u^2}{\sigma_u^2+\sigma_u^2}$ and $\sigma_* = \sqrt{\frac{\sigma_u^2\sigma_v^2}{\sigma_u^2+\sigma_v^2}}$, the unknown parameters are replaced with estimates and errors ε_i with residuals.

To conduct statistical inference using the estimated shape-aware DNN-SFM, we resort to bootstrapping. Numerical studies (see,e.g., Lakshminarayanan et al. 2017; Ovadia et al. 2019) have demonstrated that bootstrap-based uncertainty quantification methods in neural networks perform consistently well. Very recently, Padilla et al. (2024) provided finite-sample theoretical guarantees for the bootstrap procedure for constructing confidence intervals in dense ReLU networks under mild regularity assumptions. Bootstrap involves repeatedly resampling from the original dataset with replacement and re-estimating the DNN-SFM using each bootstrap sample. The resulting bootstrap distributions of the parameter estimates allow us to construct empirical confidence intervals and carry out testing.[2]

A related question is whether we can effectively separate inefficiency U from misspecification error in the DNN-SFM estimation. As in all cross-sectional stochastic frontier models, one cannot fully disentangle inefficiency from misspecification because inefficiency is identified only relative to the production frontier. For instance, any functional-form misspecification will translate into U. This limits all parametric, semiparametric, and nonparametric frontier estimators including ours. The advantage of the DNN-SFM is not that it eliminates this problem but that it reduces the scope for misspecification by approximating the production frontier within an economically admissible class of functions.

[2] In settings where wrong skewness arises in a non-negligible fraction of bootstrap replications, bootstrap-based inference may become unreliable. Practitioners should carefully inspect the full bootstrap distribution, report the frequency of boundary estimates, and interpret bootstrap confidence intervals with appropriate caution when wrong skewness is present. We did not encounter this problem in our applications of DNN-SFM.

At the same time, DNN-SFMs remain sufficiently flexible to capture nonlinearities and high-order interactions among inputs. By construction, this confines misspecification to directions that are consistent with the axioms of production. As a result, persistent deviations below the estimated frontier are less likely to reflect functional-form error and more likely to be attributable to technical inefficiency.

3.2 Model selection and hyperparameter tuning

Model selection in DNNs includes determining the appropriate number of layers, hidden nodes, and other architectural hyperparameters. We adopt a cross-validation approach to avoid overfitting while ensuring robust generalization performance. Specifically, we consider a grid of candidate architectures varying in depth ($L = 1, 2, 3$) and in width (2^2, 2^3, 2^4 and 2^5 hidden nodes per layer). For each candidate model, predictive performance was evaluated using cross-validation, with leave-one-out cross-validation (LOOCV). LOOCV ensures that each observation contributes to both training and validation, thereby maximizing the effective use of available data while still providing an unbiased estimate of the predictive error. The final model was selected as the architecture that minimized the average prediction error across the validation folds.

For the step size, we chose a between 0.1 and 0.01 using cross-validation. For the batch size, since the sample size in our simulations and application is not large, we set the batch size equal to the sample size.

3.3 Using Shapley values to interpret DNN-SFMs

DNNs are known for their high accuracy in out-of-sample predictions. However, they operate as complex, black-box systems with multiple layers of interconnected neurons, making it difficult to understand how specific inputs influence output. We follow Lundberg and Lee (2017) and use the SHAP (SHapley Additive exPlanations) values to provide a partial effect interpretation to inputs for DNN inputs. The interpretation is different from the marginal product interpretation which is familiar to economists. The difference is that it does not assume that all other factors of production remain constant. Instead, SHAP utilizes the fundamental idea from cooperative game theory proposed by Shapley (1953, 1969), which measures each player's (each input's) contribution to the overall outcome (output), considering all possible combinations of other players (other outputs).

To define SHAP formally, let $[K]$ be the set of all K inputs in the model. Let $\mathcal{S} \subseteq [K]$ denote a subset of inputs and $\mathcal{S}\backslash\{j\}$ denote the subset $\mathcal{S}$ excluding input j. Let $|\cdot|$ denote the size of a subset. Suppose that the function $\nu_{\mathcal{S}}(\boldsymbol{x})$ represents the model's expected quantity of interest (expected output) for a data point $\boldsymbol{x}$ when only the value of the inputs in $\mathcal{S}$ are fixed at their corresponding magnitudes in $\boldsymbol{x}$, while the other inputs are integrated over. The easiest way to see what this does is using the linear example $f(\boldsymbol{x}) = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_3$. Clearly, $K = 3$ and $[K] = \{1, 2, 3\}$. Suppose $\mathcal{S} = \{1, 2\}$ then $\nu_{\mathcal{S}}(\boldsymbol{x}) = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_3 E[x_3]$.

The SHAP value of input j at a given data point $\boldsymbol{x}$, denoted $\phi_j(\boldsymbol{x})$, is the weighted average of the input's contributions to $v_{\mathcal{S}}(\boldsymbol{x})$ across all possible subsets of inputs, where an input's contribution to $v_{\mathcal{S}}(\boldsymbol{x})$ is defined as the difference in $v_{\mathcal{S}}(\boldsymbol{x})$ obtained with and without input j in $\mathcal{S}$. Formally, SHAP is defined as follows (see Lundberg and Lee 2017, Theorem 1):

$$\phi_j(\boldsymbol{x}) = \frac{1}{K} \sum_{\mathcal{S} \subseteq [K]\backslash\{j\}} \binom{K-1}{|\mathcal{S}|}^{-1} \left(\nu_{\mathcal{S}\cup\{j\}}(\boldsymbol{x}) - \nu_{\mathcal{S}}(\boldsymbol{x})\right). \tag{16}$$

This definition considers all the combinations of inputs in the set $[K]\backslash\{j\}$, which is why Eq. (16) involves the combinatorial weights. Being a function of $\boldsymbol{x}$, SHAP quantifies the contribution of each input j to the output at each data point, rather than giving a global measure of input importance over all data points. As mentioned in the Introduction, the SHAP values should be understood as model-based associations, not as structural marginal products. Unlike partial derivatives of a known production function, SHAP values reflect how an input contributes to the predicted output relative to a baseline, integrating over the empirical distribution of other inputs. As a result, SHAP values generally differ from marginal effects and may be negative even when the production function is monotone. For example, the SHAP value of input j at point $\boldsymbol{x}$ for the linear model $f(\boldsymbol{x}) = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_3$ can be shown to be $\beta_j(x_j - E[x_j])$, which is different for each sample value of x_j.

Lundberg and Lee (2017) defined a class of additive input attribution methods and showed that SHAP is what they call the unique linear additive explanation. That is, SHAP can be understood and interpreted through a binary input regression. For any data point, write

$$\nu_{\mathcal{S}}(\boldsymbol{x}) = \gamma_0 + \sum_{j=1}^{K} \gamma_j \boldsymbol{z}_{\mathcal{S},j}, \tag{17}$$

where $\boldsymbol{z}_{\mathcal{S},j}$ is a binary covariate taking the value 1 if the j-th input is present in $\mathcal{S}$ and 0 otherwise. They show that SHAP ϕ_j reduces to the regression coefficient β_j even if the j-th input affects the output nonlinearly and interacts with other inputs.

In the context of our model, we define $\phi_0 = E_{\boldsymbol{X}}[\widehat{\mathrm{DNN}}(\boldsymbol{X}; \widehat{\boldsymbol{w}})]$ to be the baseline DNN prediction, i.e., the expected output over the entire distribution of

inputs, where $\widehat{DNN}(\boldsymbol{X};\widehat{\boldsymbol{w}})$ is a trained DNN-SFM with weights $\widehat{\boldsymbol{w}}$, and we let

$$\nu_S(\boldsymbol{x}) = E_{\boldsymbol{X}_{[K]\setminus S}}[\widehat{\mathrm{DNN}}(\boldsymbol{X};\widehat{\boldsymbol{w}})|X_S = \boldsymbol{x}_S] \tag{18}$$

$$\widehat{DNN}(\boldsymbol{x};\widehat{\boldsymbol{w}}) = \phi_0 + \sum_{k=1}^{K}\phi_k. \tag{19}$$

We compute the SHAP values using the Python package `shap`, which implements various methods for approximating SHAP from a DNN. The method used in our implementation is the Kernel SHAP method, a model-agnostic approach based on the so-called Shapley regression. This method estimates SHAP using a weighted linear regression, where different subsets of inputs are sampled, and their contributions to the model's prediction are evaluated. An interesting theoretical extension, which we leave for future work, is to derive statistical properties of the SHAP estimates based on a trained DNN.

4 Simulation

In this section, we conduct a simulation study to assess the relative performance of the proposed estimators using DGPs similar to those used in Examples 1 and 2. Specifically, we use a standard single-input Cobb-Douglas specification, which is linear in logs, but, in addition, we consider a case with irrelevant regressors and a case with the Translog specification. For each DGP, we simulate $n = 200$ observations and repeat the experiments 100 times.

We generate the data as follows:

$$\text{DGP1}: \quad Y_i = \beta_0 X_{1i}^{\beta_1}\exp(V_i - U_i), \qquad i = 1,\ldots,n,$$

where $V_i \sim \mathrm{N}(0,\sigma_v^2)$ and $U_i \sim \mathrm{HN}(0,\sigma_u^2)$ with $\sigma_v = 0.9$ and $\sigma_u = 0.2$, $X_{1i} \sim$ Unif(0, 3) and $(\beta_0,\beta_1) = (2, 0.2)$.

In addition, we consider two more DGPs. The first one generates the data as follows:

$$\text{DGP2}: \quad Y_i = \beta_0 X_{1i}^{\beta_1} X_{2i}^{\beta_2} X_{3i}^{\beta_3} X_{4i}^{\beta_4}\exp(V_i - U_i),$$

where $\beta_0 = 2, \beta_1 = 0.3$, and $\beta_2 = \beta_3 = \beta_4 = 0$, $V_i \sim \mathrm{N}(0,\sigma_v^2)$, with $\sigma_v = 1$, and $U_i \sim \mathrm{HN}(0,\sigma_u^2)$ with $\sigma_u = 0.5$, $X_{1i} \sim$ Unif(. 5, 3.5), $X_{2i} \sim$ Unif(1.5, 4.5), $X_{3i} \sim$ Unif(2.5, 5.5), $X_{4i} \sim$ Unif(3.5, 6.5), $X_{5i} \sim$ Unif(4.5, 7.5). Clearly, X_{2i}, X_{3i} and X_{4i} are irrelevant in DGP2. However, in the estimation of DGP2 we will assume (incorrectly) that the true model contains these regressors in order to assess how the various estimators behave in the presence of irrelevant regressors.

The second DGP simulates the data using the Translog production function:

$$\text{DGP3}: Y_i = \beta_0 X_{1i}^{\beta_1} X_{2i}^{\beta_2}\exp\left[\frac{1}{2}\beta_{11}(\ln X_{1i})^2 + \frac{1}{2}\beta_{22}(\ln X_{2i})^2 + \beta_{12}\ln X_{1i}\ln X_{2i}\right]\exp(V_i - U_i),$$

where $\beta_0 = 1.0$, $\beta_1 = \beta_2 = 0.4$, $\beta_{11} = \beta_{22} = -0.35$, and $\beta_{12} = 0.15$, $V_i \sim \mathrm{N}(0,\sigma_v^2)$, with $\sigma_v = 0.8$ and $U_i \sim \mathrm{HN}(0,\sigma_u^2)$ with $\sigma_u = 0.5$, and input variables X_1 and X_2 are independently drawn from a uniform distribution on [0.05, 10]. In the estimation, we consider two cases for DGP3: one is where we assume the correct specification and the other is where we assume (incorrectly) that the true DGP is Cobb-Douglas with two inputs, X_1 and X_2. This allows us to compare the behavior of our estimators with that of MLE under an incorrect functional form.

For the estimation we use logarithms of the inputs and output and consider five types of estimators: (i) a conventional SFM (denoted by *Lin* for DGP1 and DGP2, and by *Translog* for DGP3) estimated by MLE (Aigner et al. 1977); (ii) a semiparametric generalized additive SFM (*GAM-SFM*) (Ferrara and Vidoli 2017); (iii) a Bayesian additive regression tree SFM (*BART-SFM*) (Wei et al. 2024); (iv) two unconstrained DNN models, denoted by *NN-Relu* and *NN-Elu*, which employ the standard activation functions ReLU and ELU, respectively; and (v) two shape-constrained DNN models, using the CReLU and CELU activation functions from Section 2.3, denoted as *SNN-CRelu* and *SNN-CElu*, respectively. For DGP3, we also report the results obtained under the incorrect Cobb-Douglas specification. These results are denoted by *Misspec (Lin)*.

The GAM-SFM fits the following model of the conditional expectation of log-output:

$$E(\log Y|\boldsymbol{X} = \boldsymbol{x}) = \psi_0 + \sum_{j=1}^{k}\psi_j(\log x_j), \tag{20}$$

where $\psi_j(\cdot)$'s are smooth basis functions. The model satisfies monotonicity constraints by means of P-splines and is estimated by pseudo-MLE using the R package `semsfa`(see Ferrara and Vidoli 2021).

The BART-SFM fits the model

$$E(\log Y|\boldsymbol{X} = \boldsymbol{x}) = \beta_0 + \sum_{j=1}^{m} g_j(\log \boldsymbol{x}; T_j, M_j), \tag{21}$$

where $g_j(\cdot\,; T_j, M_j)$ is a weak learner function, represented by a binary decision tree T_j and a vector M_j, which collects

individual parameters of the tree leaves. It is estimated using *R* with 4,000 MCMC samples, where convergence is assessed using trace plots for σ_u and σ_v and Geweke's convergence diagnostics (Geweke 1992). The trace plots indicated that a burn-in of 1,000 samples was acceptable for all cases and that the chains were mixing well. All of Geweke's convergence diagnostics had values that indicate convergence.

We report RMSE of the estimated production function, average bias of $\hat{\sigma}_v$ and $\hat{\sigma}_u$, and average bias in technical inefficiency scores. The four performance metrics are computed as follows:

$$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}\left(\frac{\hat{Y}_i - Y_i}{Y_i}\right)^2},\ \text{Bias}_v = |\hat{\sigma}_v - \sigma_v|,$$
$$\text{Bias}_u = |\hat{\sigma}_u - \sigma_u|,\ \text{Bias}_{\text{TE}} = \frac{1}{n}\sum_{i=1}^{n}|\hat{\text{TE}}_i - \text{TE}_i|,$$

where TE_i is computed using the true parameter values and Eq. (15). Note that all the estimators produce fitted values for $\log Y$, not Y, so to obtain the fitted values for Y we exponentiate the fitted values for $\log Y$ and correct them using the mean exponent of residuals. The performance metrics of BART-SFM are calculated based on the posterior modes in order to make them more comparable with a penalized MLE estimator.

The simulation results are summarized in Table 1. Several observations are worth highlighting. First, the shape-constrained estimators *SNN-CRelu* and *SNN-CElu* perform better than all other estimators in terms of almost all performance metrics. Besides *Lin* and *Translog*, the only exception is Bias_v in DGP3 which is slightly lower for *GAM-SFM*, *BART-SFM* and even *Misspec (Lin)* than for *SNN-CRelu* and *SNN-CElu*, but this is offset by a much better performance of *SNN-CRelu* and *SNN-CElu* in terms of Bias_u, RMSE and RMSE_{TE}. The unchallenged performance of *Lin* and *Translog* is not surprising given that these are parametric estimators under correct specifications. It is perhaps interesting to observe that *Lin* shows the leading performance in DGP2 even in the presence of irrelevant regressors. For DGP3, *Misspec (Lin)* performs much worse overall, being an estimator based on an incorrectly specified model, however, both *GAM-SFM* and *BART-SFM* performs worse than *Misspec (Lin)* by a number of metrics.

Second, the four DNN-based estimators dominate the nonparametric and Bayesian alternatives (*GAM-SFM* and *BART-SFM*) for DGP1. This is true whether or not the DNN learning is shape-aware. In the presence of irrelevant regressors (DGP2) the situation changes. For DGP2, the standard DNN estimators (*NN-Relu* and *NN-Elu*) visibly drop in all the metrics, while the shape-aware estimators (*SNN-CRelu* and *SNN-CElu*) still show the leading performance among the nonparametric and Bayesian alternatives. Similarly for

Table 1 RMSE and biases of estimators

DGP1

				unconstrained		shape-constrained	
	Lin	GAM-SFM	BART-SFM	NN-Relu	NN-Elu	SNN-CRelu	SNN-CElu
RMSE	0.277	1.8863	1.0235	0.622	0.583	0.510	0.518
Bias_v	0.007	0.0787	0.0402	0.060	0.049	0.035	0.036
Bias_u	0.020	0.3550	0.1129	0.013	0.011	0.008	0.008
Bias_{TE}	0.023	0.2052	0.0807	0.039	0.038	0.035	0.035
RMSE_{TE}	0.020	0.2141	0.0895	0.041	0.040	0.037	0.037

DGP2

				unconstrained		shape-constrained	
	Lin	GAM-SFM	BART-SFM	NN-Relu	NN-Elu	SNN-CRelu	SNN-CElu
RMSE	0.195	0.8595	0.3873	1.063	1.025	0.394	0.406
Bias_v	0.009	0.1535	0.0955	0.585	0.512	0.054	0.060
Bias_u	0.020	0.3129	0.1237	0.219	0.192	0.020	0.023
Bias_{TE}	0.031	0.1857	0.0899	0.145	0.125	0.028	0.028
RMSE_{TE}	0.017	0.1857	0.0959	0.145	0.126	0.033	0.034

DGP3

					unconstrained		shape-constrained	
	Misspec (Lin)	Translog	GAM-SFM	BART-SFM	NN-Relu	NN-Elu	SNN-CRelu	SNN-CElu
RMSE	1.006	0.312	0.974	0.483	0.404	0.431	0.347	0.347
Bias_v	0.073	0.068	0.083	0.071	0.144	0.157	0.084	0.086
Bias_u	0.390	0.300	0.310	0.337	0.090	0.098	0.053	0.054
Bias_{TE}	0.157	0.124	0.160	0.203	0.075	0.079	0.055	0.056
RMSE_{TE}	0.168	0.130	0.173	0.217	0.079	0.084	0.057	0.058
Runtime (sec)	0.020	0.031	0.048	78.631	1.06	1.10	1.20	5.52

DGP3, the shape-aware estimators have visibly better performance than the standard ones.

The improved performance may come at the expense of computational time. We report the average runtimes (in seconds) of the various estimators for DGP3 in the last line of Table 1. The runtimes are calculated as averages over 10 replications of each method. *BART-SFM* and *GAM-SFM* are fitted in `R`, while all the other estimators are obtained in `Python`. The DNN models are trained in `PyTorch` using the `Adam` optimizer with up to 2,500 epochs. Convergence was monitored through the loss trajectory, and in all replications the loss stabilized well before reaching the maximum number of epochs, indicating stable optimization behavior.

As can be seen from the runtimes, the DNN estimators are noticeably more demanding computationally than the other non-Bayesian estimators, yet remain reasonably fast and affordable. *SNN-CRelu* takes approximately as long as *NN-Relu* and *NN-Elu*. It is perhaps surprising that *SNN-CElu* takes about 5 times longer which may have to do with its nonlinearity.

It also helps to have some measures of the extent to which the different estimators are subject to the "wrong skewness" problem. We investigate this using DGP3 and several measures obtained using the same simulated data as before. First, we counted the number of replications (out of 100) when the OLS regression using DGP3 produces positively skewed residuals. There were 14 instances. Second, we did the same for the unconstrained DNN which uses ReLU and the MSE loss, rather than ReLU and likelihood loss. This isolates the effect of using DNN on the same loss as OLS. That number was 10 (out of 100). Third, we counted the instances of convergence failure due to vanishing σ_u^2 for the regular MLE in the R package `frontier` (the *Translog* estimator) and identified the same 14 instances. Fourth, for *GAM-SFM*, we observed that the R package `semsfa` also produced a wrong skew warning for the same 14 data sets. Fifth, we used the unconstrained DNN with the ReLU and ELU activations and the likelihood loss (the *NN-Relu* and *NN-Elu* estimators) and obtained *zero* instances of wrong skew or convergence issues. Finally, we used the constrained DNN with CReLU and CELU and the likelihood loss (the *SNN-CRelu* and *SNN-CElu* estimators) and, again, obtained *zero* instances of wrong skew or convergence problems.

Overall, these experiments suggest the the wrong skew problem is less prevalent in likelihood-based DNN estimators than in any other non-Bayesian estimator we consider. We attribute this to the flexibility of DNNs combined with the structural requirements imposed by the likelihood.

5 Empirical application

In this section, we apply DNN-SFM to a real data set collected from $n = 43$ small holder rice producers in the Tarlac region of the Philippines in 1992. This is a subset of a benchmark data set used in numerous studies before, but usually as a panel consisting of eight years of data, not one year (see, e.g., Coelli et al. 2005; Henningsen 2014). The output variable Y is tonnes of freshly threshed rice for each farm, and the three input variables are area planted in hectares (Area), man-days of family and hired labor (Labor), and fertilizer used in kg of active ingredients (NPK).

We fit the linear SFM, BART, NN-ELU, and SNN-CELU to the data. To select the neural network architecture, we use two-layer networks with the following candidate number of nodes in each layer: [32,16], [32,8], [16,8], [8,4], and [4,4]. Leave-one-out cross-validation (LOOCV) is applied to evaluate the DNN predictive performance. The [32,16] architecture provided the lowest LOOCV loss and was therefore selected. For the linear SFM we use the `sfa` function in the R package `frontier`(Coelli et al. 2013). For BART-SFM, we use the R code from Wei et al. (2024). All estimations are in logs, not levels.

The resulting efficiency scores and their summary statistics are shown in Table 2. We note that the linear SFM estimated using the `frontier` package returned a warning message that the model is misspecified and the estimate of σ_u is zero, failing to calculate the technical inefficiency scores. An examination of OLS residuals revealed a strong positive skewness indicating the "wrong skewness" problem. A similar problem using this data was found by other people. For example, Wang et al. (2011) rejected the normal/half-normal distributional assumption for this data in the conventional linear specification.

Neither BART nor NN-Elu or SNN-CElu suffer from this problem and produce both TE scores and ranks. Overall, the BART efficiencies are higher and have a narrower range than those of NN-Elu and SNN-CElu, while the shape-aware efficiencies (SNN-CElu) are lower and have a wider range than shape-agnostic (NN-Elu). The farm rankings produced by the three estimators are quite different as illustrated by the ranks (and efficiency scores) of the top and bottom five farms reported in the right panel of Table 2. In fact, Kendall's τ's for the estimator pairs are τ(BART,NN) = 0.464, τ(BART,SNN) = 0.743 and τ(NN,SNN) = 0.464, where the strongest association between SNN-CELU and BART indicates a high degree of concordance between the efficiency scores. This is likely due to the fact that, unlike NN-Elu, BART imposes monotonicity, while SNN-CElu imposes both monotonicity and concavity.

Next we implement bootstrap resampling in order to quantify uncertainty associated with the SNN-CElu estimation.

Table 2 The descriptive statistics and top/bottom ranks of Philippines rice farm technical efficiency scores

					Five most efficient			Five least efficient		
	MLE	BART	NN-Elu	SNN-CElu	BART	NN-Elu	SNN-CElu	BART	NN-Elu	SNN-CElu
Mean	-	0.9721	0.9508	0.9106	90	114	90	126	89	97
S.D.	-	0.0023	0.0142	0.0293	(0.9768)	(0.9737)	(0.9615)	(0.9652)	(0.9073)	(0.8376)
Min	-	0.9666	0.9073	0.8376	104	117	117	94	126	120
Max	-	0.9768	0.9737	0.9615	(0.9762)	(0.9683)	(0.9473)	(0.9662)	(0.9101)	(0.8415)
					123	110	123	129	94	94
					(0.9754)	(0.9682)	(0.9443)	(0.9672)	(0.9225)	(0.8426)
					117	128	123	97	99	126
					(0.9752)	(0.9671)	(0.9407)	(0.9674)	(0.9315)	(0.8489)
					127	127	118	120	118	129
					(0.9746)	(0.9645)	(0.9405)	(0.9676)	(0.9323)	(0.8656)

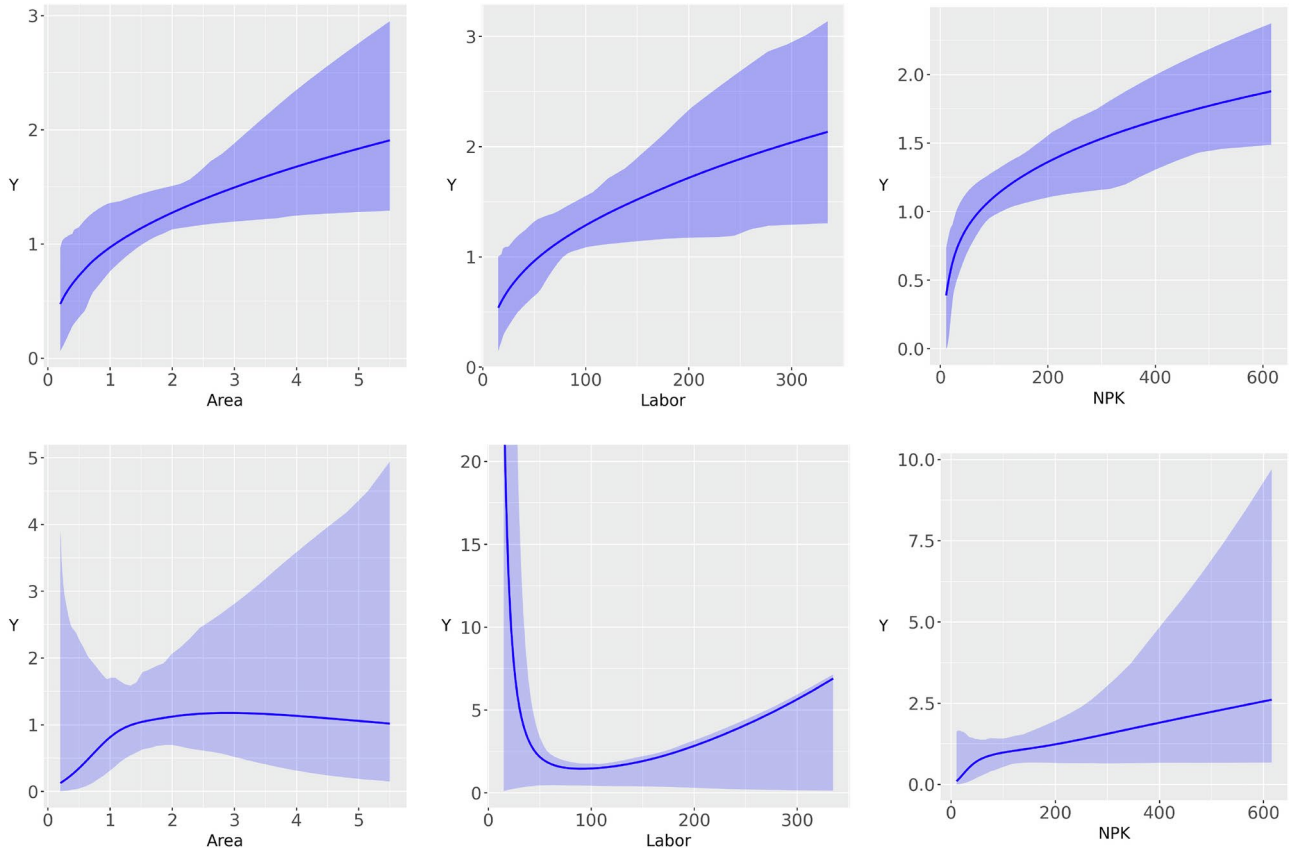

Fig. 4 Partial dependence plots (solid curve) with 95% bootstrap confidence intervals (shaded area) for SNN-CElu (top panels) and NN-Elu (bottom panels)

Specifically, we estimate the variability in the estimation of σ_v (the noise component) and σ_u (the inefficiency component) of the error, as well as draw partial dependence curves for inputs. The estimated values are $\hat{\sigma}_v = 0.161$ with a 95% bootstrap confidence interval of $(0.11, 0.19)$, and $\hat{\sigma}_u = 0.112$ with a 95% bootstrap confidence interval of (0.076, 0.132). For reference, the posterior mode estimates from BART-SFM are $\hat{\sigma}_v = 0.175$ with a 95% highest density interval (HDI) of (0.134, 0.238), and $\hat{\sigma}_u = 0.029$ with a 95% HDI of (0.017, 0.061). Figure 4 plots the partial dependence curves of the three inputs for SNN-CElu and NN-Elu. Two things are apparent from the plots. First, SNN-CElu reflects decreasing marginal products for all the inputs and is generally subject to greater statistical uncertainty at higher values of inputs. Second, using the shape-agnostic DNN (NN-Elu) results in a non-monotone production function with respect to two out of the three inputs in this example.

For completeness, Fig. 5 provides kernel density estimates for efficiency scores from the three models, namely SNN-CElu, NN-Elu and BART-SFM. It is clear from the

figure that SNN-CElu shows a trimodal distributional pattern that is not detected by the other two approaches. The SNN-CElu fit suggests that farms can be classified into low-, medium- and high-efficiency farms, with the high-efficiency group being most prevalent. The other two approaches produce significantly different density estimates, unimodal in both cases, with the BART-SFM efficiency scores being generally higher than both NN-Elu and SNN-CElu.

Finally, we evaluate the SHAP values for this empirical example using SHAP value plots and a SHAP dependence plot. Figure 6 presents the SHAP value plots for the SNN-CElu (upper plot) and NN-ELu model (lower plot). Positive SHAP values indicate farms for which freeing input *j* from the farm's current value increases the predicted output, while negative SHAP values indicate farms for which the predicted output would go down if we free up the farm's current input *j*. The top panel shows that, due to the shape constraints imposed by SNN-CElu, the SHAP values grow in a monotone fashion with the magnitude of each input, color-coded from blue (low) to red (high). Each dot on these plots corresponds to a farm, so the plots provide a detailed view of input importance and contribution at each data point. The plot, produced using the Python package `shap`, automatically places on top the input with the highest absolute SHAP values (Labor in the upper plot).

The lower panel of Fig. 6 shows that the unconstrained estimation violates the property of monotonicity and concavity of SHAP with respect to Labor. This is apparent from the mixing of points with high and low Labor (and contrasting colors) in the regions with both positive and negative SHAP. At the same time, higher values of Labor generally coincide with negative SHAP suggesting an inverse relationship between labor and its contribution to output. Interestingly, in the shape-agnostic model, the Labor variable has the smallest contribution to the output as measured by the mean absolute SHAP value.

Figure 6 serves two practical purposes. First, it provides a diagnostic check that the shape restrictions translate into economically sensible local input contributions. In the shape-aware model (SNN-CElu), SHAP values increase monotonically with input levels, reflecting free disposability and diminishing marginal returns at the level of individual farms, while the unconstrained model does not do that. Second, Fig. 6 highlights heterogeneity in input contributions across farms, even when the global production frontier is concave. Farms with similar input

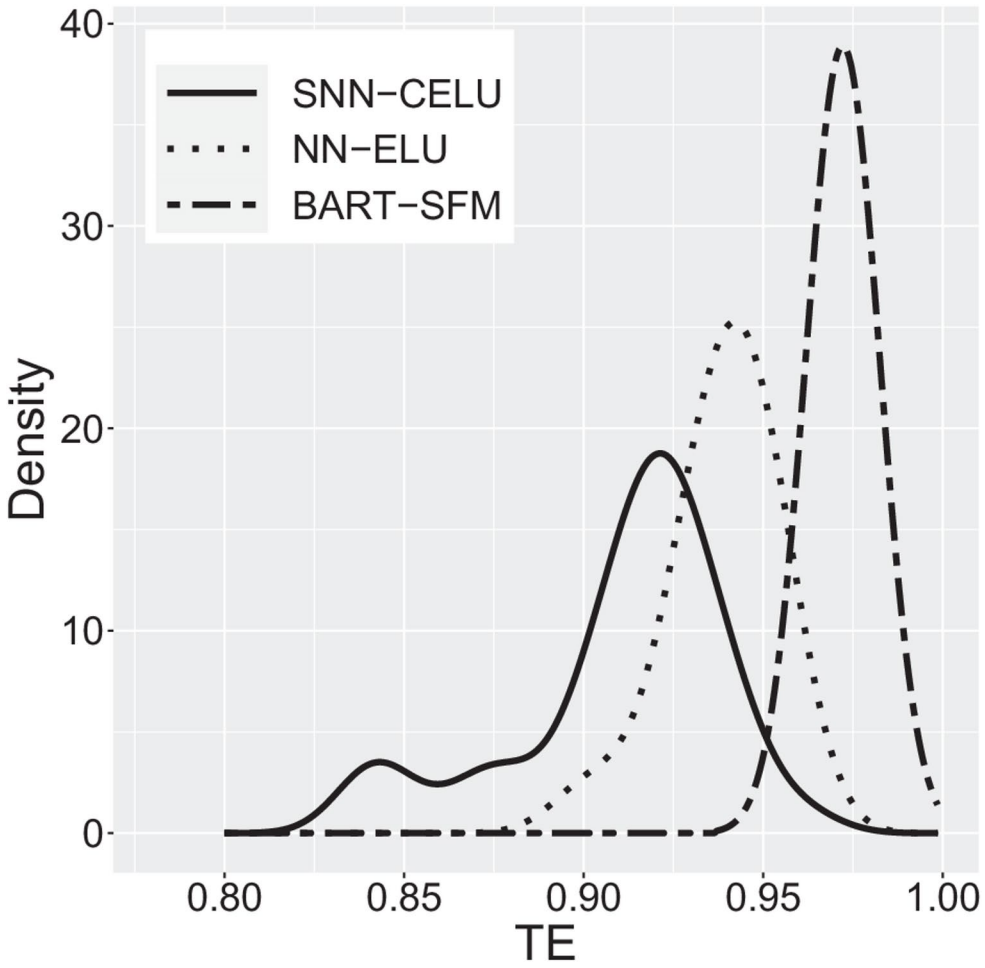

Fig. 5 Kernel density estimates for efficiency scores from SNN-CElu (solid), NN-ELU(dotted) and BART-SFM(dot-dashed)

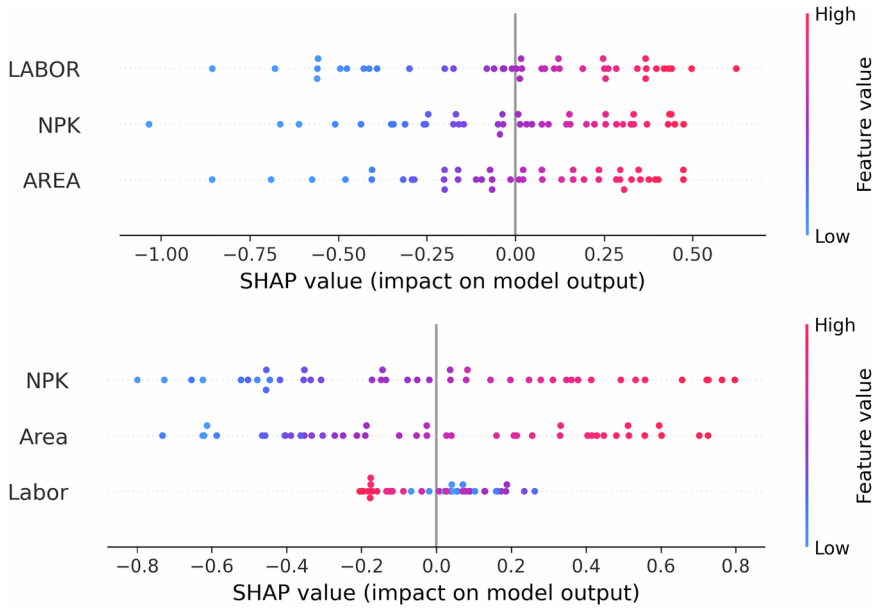

Fig. 6 SHAP summary plots for SNN-CElu (upper plot) and NN-Elu (lower plot) models

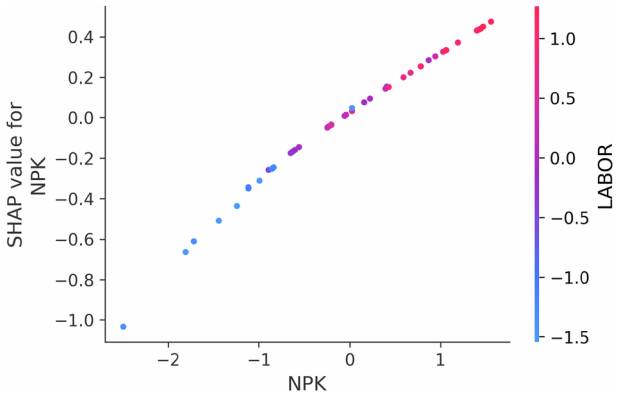

Fig. 7 SHAP dependence plot for NPK and Labor based on SNN-CElu

levels have very different SHAP values. The observation-specific interpretation may be more practical than reliance on average partial effects.

Figure 7 presents a SHAP dependence plot from the SNN-RElu model for the two inputs with the highest contribution to farm production, namely, NPK and Labor. The curved relationship between NPK and its SHAP value reflects diminishing marginal contributions of fertilizer, while the color gradient reveals that the contribution of NPK depends systematically on labor intensity. This interaction would be difficult to detect using linear, translog, or additive frontier specifications. From an applied perspective, this figure helps explain why farms with similar fertilizer usage can exhibit different predicted outputs and efficiency levels: differences in complementary inputs, such as labor, alter the marginal contribution of fertilizer.

6 Conclusion

This paper introduced shape-aware deep learning to stochastic frontier models in a way that imposes monotonicity and concavity to ensure economic interpretability while leveraging the flexibility of deep neural networks. The proposed method (DNN-SFM) demonstrates superior predictive accuracy and robustness in simulations. Our simulation experiments and application to real-world production data illustrate its ability to overcome such issues as model misspecification and the "wrong skew" problem.

The applied SFA community cares deeply about explaining inefficiency, not just estimating it. DNN-SFM is compatible with inefficiency heterogeneity, modeled, for example, by making the distributional parameters of U_i depend on observed covariates. In particular, one may let σ_u^2 depend on a vector of variables Z_i, $\sigma_u^2 = \exp(g(Z_i))$, where $g(\cdot)$ is a flexible function approximated by a DNN. Under this formulation, estimation proceeds by jointly training two DNNs, one of which is shape-aware, while the other is generic. We leave a full investigation of heteroskedastic DNN-SFMs for future research.

Another exciting direction for future research is to extend DNN-SFM to accommodate panel and spatial data structures, allowing for modeling time-varying inefficiencies, dynamic productivity changes and geographic spillover effects (Parmeter and Kumbhakar 2025). Further developing uncertainty qualification for DNN-based SFMs, in particular for SHAP values, is also promising. The relevant methodology for this may come from the union of such areas as Bayesian neural network modelling (see, e.g., Papamarkou et al. 2024), model-agnostic conformal inference (see, e.g., Angelopoulos et al. 2023), neural network

ensemble learning (see, e.g., Lakshminarayanan et al. 2017) and Monte Carlo dropouts (see, e.g., Gal and Ghahramani 2016).

Acknowledgements The research of Zheng Wei was supported by the Hispanic Serving Institutions Education Grants Program, Grant no:2023-77040-41202, from the U.S. Department of Agriculture (USDA), National Institute of Food and Agriculture (NIFA). The research of Artem Prokhorov was supported by the Basic Research Program of HSE University, Center for Big Data in Economics and Finance. The research of Huiyan Sang was supported by NSF grant no. NSF DMS-2220231 and NSF SES-2521573. We sincerely thank the Editor, Professor Christopher Parmeter, for his insightful comments, which greatly improved this paper.

Author contributions All authors wrote and edited the main manuscript text. Z.W. developed computer code for simulation and real data analysis.

Funding Open access funding provided by Texas A&M University-Corpus Christi.

Data Availability The codes and data used for simulations and application are available in Python on the authors' web pages and at a GitHub repository:https://github.com/ZWeiSTAT/shape-constrained_neural_networks

Compliance with ethical standards

Conflict of interest The authors declare no conflict of interest.

References

Aigner D, Lovell CK, Schmidt P (1977) Formulation and estimation of stochastic frontier production function models. Journal of Econometrics 6:21–37

Angelopoulos AN, Bates S (2023) Conformal prediction: A gentle introduction. Foundations and Trends ® in Machine Learning 16:494–591

Bishop, C. M. and Bishop, H. (2023): *Deep learning: Foundations and concepts*, Springer Nature.

Boyd S, Vandenberghe L (2004) Convex Optimization. Cambridge University Press, Cambridge, UK

Clevert, D.-A., Unterthiner, T. and Hochreiter, S. (2016): Fast and Accurate Deep Network Learning by Exponential Linear Units (ELUs), in *International Conference on Learning Representations (ICLR)*.

Coelli, T., Henningsen, A. and Henningsen, M. A. (2013): *Package 'frontier': Stochastic Frontier Analysis*, R package version 1.1.

Coelli, T. J., Rao, D. S. P., O'Donnell, C. J. and Battese, G. E. (2005): *An Introduction to Efficiency and Productivity Analysis*, Boston, MA: Springer US, 2nd ed.

Cybenko G (1989) Approximation by superpositions of a sigmoidal function. Mathematical Control Signal Systems 2:303–314

DeVore R, Hanin B, Petrova G (2021) Neural network approximation. Acta Numerica 30:327–444

Fan Y, Li Q, Weersink A (1996) Semiparametric estimation of stochastic production frontier models. Journal of Business & Economic Statistics 14:460–468

Ferrara G, Vidoli F (2017) Semiparametric stochastic frontier models: A generalized additive model approach. European Journal of Operational Research 258:761–777

——— (2021): semsfa: Semiparametric Estimation of Stochastic Frontier Models, R package version 1.1, retrieved from https://CRAN.R-project.org/package=semsfa

Gal, Y. and Ghahramani, Z. (2016): Dropout as a bayesian approximation: Representing model uncertainty in deep learning, in *International Conference on Machine Learning*, PMLR, 1050–1059.

Geweke, J. (1992): Evaluating the accuracy of sampling-based approaches to the calculation of posterior moments, in *Bayesian Statistics 4*, ed. by J. M. Bernardo, J. O. Berger, A. P. Dawid, and A. F. M. Smith, Oxford: Oxford University Press, 169-193.

Giannakas K, Tran KC, Tzouvelekas V (2003) On the choice of functional form in stochastic frontier modeling. Empirical Economics 28:75–100

Goodfellow, I., Bengio, Y., Courville, A. and Bengio, Y. (2016): *Deep learning*, vol. 1, MIT press Cambridge.

Hanson DL, Pledger G (1976) Consistency in Concave Regression. The Annals of Statistics 4:1038–1050

He, K., Zhang, X., Ren, S. and Sun, J. (2015): Delving deep into rectifiers: Surpassing human-level performance on imagenet classification, in *Proceedings of the IEEE International Conference on Computer Vision*, 1026–1034.

Henningsen, A. (2014): Introduction to econometric production analysis with R, *Department of Food and Resource Economics, University of Copenhagen*.

Hornik K (1991) Approximation capabilities of multilayer feedforward networks. Neural Networks 4:251–257

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

Kumbhakar SC, Lovell CK (2003) Stochastic Frontier Analysis. Cambridge University Press, Cambridge, UK

Kumbhakar SC, Park BU, Simar L, Tsionas EG (2007) Nonparametric stochastic frontiers: A local maximum likelihood approach. Journal of Econometrics 137:1–27

Kuosmanen T, Kortelainen M (2012) Stochastic non-smooth envelopment of data: Semi-parametric frontier estimation subject to shape constraints. Journal of Productivity Analysis 38:11–28

Kutlu, L. and Mao, X. (2024): A Machine Learning Approach to Stochastic Frontier Modeling, *Preprint*.

Kutlu L, Tran KC, Tsionas MG (2020) A spatial stochastic frontier model with endogenous frontier and environmental variables. European Journal of Operational Research 286:389–399

Lakshminarayanan, B., Pritzel, A. and Blundell, C. (2017): Simple and scalable predictive uncertainty estimation using deep ensembles, *Advances in Neural Information Processing Systems*, 30.

LeCun Y, Bengio Y, Hinton G (2015) Deep learning. Nature 521:436–444

LeCun Y, Boser B, Denker JS, Henderson D, Howard RE, Hubbard W, Jackel LD (1989) Backpropagation applied to handwritten zip code recognition. Neural Computation 1:541–551

Lundberg, S. M. and Lee, S.-I. (2017): A unified approach to interpreting model predictions, in *Proceedings of the 31st International Conference on Neural Information Processing Systems*, Red Hook, NY, USA: Curran Associates Inc., NIPS'17, 4768–4777.

Maas, A. L., Hannun, A. Y. and Ng, A. Y. (2013): Rectifier Nonlinearities Improve Neural Network Acoustic Models, in *Proceedings of the 30th International Conference on Machine Learning (ICML)*, JMLR Workshop and Conference Proceedings, ICML '13, 3–8.

Meeusen, W. and van Den Broeck, J. (1977): Efficiency estimation from Cobb-Douglas production functions with composed error, *International Economic Review*, 435–444.

Nair, V. and Hinton, G. E. (2010): Rectified linear units improve restricted boltzmann machines, in *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, 807–814.

Ovadia, Y., Fertig, E., Ren, J., Nado, Z., Sculley, D., Nowozin, S., Dillon, J., Lakshminarayanan, B. and Snoek, J. (2019): Can you trust your model's uncertainty? evaluating predictive uncertainty under dataset shift, *Advances in neural information processing systems*, 32.

Padilla, C. M. M., Padilla, O. H. M., Kei, Y. L., Zhang, Z. and Chen, Y. (2024): Confidence interval construction and conditional variance estimation with dense relu networks, *arXiv preprint arXiv:2412.20355*.

Papamarkou, T., Skoularidou, M., Palla, K., Aitchison, L., Arbel, J., Dunson, D., Filippone, M., Fortuin, V., Hennig, P., Hernández-Lobato, J. M. et al. (2024): Position: Bayesian deep learning is needed in the age of large-scale AI, *arXiv preprint arXiv:2402.00809*.

Parmeter, C. F. and Kumbhakar, S. C. (2025): The generalized panel data stochastic frontier model: A review and nonparametric estimation, *Journal of Productivity Analysis*, 1–19.

Parmeter, C. F. and Racine, J. S. (2012): Smooth Constrained Frontier Analysis, in *Recent Advances and Future Directions in Causality, Prediction, and Specification Analysis*, ed. by X. Chen and N. R. Swanson, New York: Springer, 463–488.

Parmeter CF, Wang H-J, Kumbhakar SC (2017) Nonparametric estimation of the determinants of inefficiency. Journal of Productivity Analysis 47:205–221

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L. et al. (2019): Pytorch: An imperative style, high-performance deep learning library, *Advances in Neural Information Processing Systems*, 32.

Poggio T (2017) Why and when can deep—but not shallow—networks avoid the curse of dimensionality. International Journal of Automation and Computing 14:503–519

Prokhorov, A. (2024): *Efficiency and productivity analysis: Using copulas in stochastic frontier models*, Taylor & Francis.

Ramachandran, P., Zoph, B. and Le, Q. V. (2017): Searching for activation functions, *arXiv preprint arXiv:1710.05941*.

Rumelhart DE, Hinton GE, Williams RJ (1986) Learning representations by back-propagating errors. Nature 323:533–536

Shapley, L. S. (1953): A value for n-person games, *Contribution to the Theory of Games*, 2.

——— (1969): Utility comparison and the theory of games, *The Shapley Value. Essays in Honor of Lloyd S. Shapley*, 307–319.

Sonoda S, Murata N (2017) Neural network with unbounded activation functions is universal approximator. Applied and Computational Harmonic Analysis 43:233–268

Tran KC, Tsionas MG, Prokhorov AB (2023) Semiparametric estimation of spatial autoregressive smooth-coefficient panel stochastic frontier models. European Journal of Operational Research 304:1189–1199

Tsionas M, Parmeter CF, Zelenyuk V (2023) Bayesian artificial neural networks for frontier efficiency analysis. Journal of Econometrics 236:105491

Tsionas MG (2020) Quantile stochastic frontiers. European Journal of Operational Research 282:1177–1184

——— (2022a): Efficiency estimation using probabilistic regression trees with an application to Chilean manufacturing industries, *International Journal of Production Economics*, 108492.

———(2022b) Estimating monotone concave stochastic production frontiers. *Journal of Business & Economic Statistics*, **40**, 1403–1414.

Tsionas MG, Mamatzakis E (2019) Further results on estimating inefficiency effects in stochastic frontier models. European Journal of Operational Research 275:1157–1164

Van Rossum G, Drake FL (2009) Python 3 Reference Manual. CreateSpace, Scotts Valley, CA

Wang S (1996) Nonparametric econometric modelling: A neural network approach. European Journal of Operational Research 89:581–592

——— (2003): Adaptive non-parametric efficiency frontier analysis: A neural-network-based model, *Computers & Operations Research*, **30**, 279–295.

Wang WS, Amsler C, Schmidt P (2011) Goodness of fit tests in stochastic frontier models. Journal of Productivity Analysis 35:95–118

Wei, Z., Sang, H. and Coulibaly, N. (2024): Nonparametric Machine Learning for Stochastic Frontier Analysis: A Bayesian Additive Regression Tree Approach, *Econometrics and Statistics*.

Yarotsky D (2017) Error bounds for approximations with deep ReLU networks. Neural Networks 94:103–114

Zellner, A., Kmenta, J. and Dreze, J. (1966): Specification and estimation of Cobb-Douglas production function models, *Econometrica: Journal of the Econometric Society*, 784–795.