Spring 2024

Scaled and Graduated Learning in Deep RELU Networks and Reconstructing Depp Inelastic Scattering Kinematics

Abdullah Ayar Farhat
Old Dominion University, afarh002@odu.edu

Recommended Citation

Farhat, Abdullah A.. "Scaled and Graduated Learning in Deep RELU Networks and Reconstructing Depp Inelastic Scattering Kinematics" (2024). Doctor of Philosophy (PhD), Dissertation, Mathematics & Statistics, Old Dominion University, DOI: 10.25777/nvkw-dk64
https://digitalcommons.odu.edu/mathstat_etds/127

SCALED AND GRADUATED LEARNING IN DEEP RELU NETWORKS AND RECONSTRUCTING DEEP INELASTIC SCATTERING KINEMATICS

by

Abdullah Ayar Farhat
B.A. May 2013, Johns Hopkins University
M.A. May 2013, Johns Hopkins University

A Dissertation Submitted to the Faculty of
Old Dominion University in Partial Fulfillment of the
Requirements for the Degree of

DOCTOR OF PHILOSOPHY

COMPUTATIONAL AND APPLIED MATHEMATICS

OLD DOMINION UNIVERSITY
May 2024

ABSTRACT

SCALED AND GRADUATED LEARNING IN DEEP RELU NETWORKS AND RECONSTRUCTING DEEP INELASTIC SCATTERING KINEMATICS

Abdullah Ayar Farhat
Old Dominion University, 2024
Director: Dr. Yuesheng Xu

To address computational challenges in learning deep neural networks, properties of deep RELU networks were studied to develop a multi-scale learning model. The multi-scale model was compared to the multi-grade learning models. Unlike the deep neural network learned from the standard single-scale, single-grade model, the multi-scale neural networks use low scale information from all hidden layers, and thusly provide a robust approximation method that requires fewer parameters, lower computational time, and is resistant to noise. It is shown that the multi-scale method is not subject to issues arising from the vanishing gradient problem. This allows very deep multi-scale networks to be effectively trained. It is proven that the collection of multi-scale neural networks are universal approximators in the space of continuous functions. The neural network learned from a multi-grade model is the superposition of the neural networks, in a stair-shape, each of which is learned from one grade of the learning. Three proof-of-concept numerical examples presented in the paper demonstrate that the multi-scale and multi-grade methods are superior to the single-scale, single-grade networks. The extended analysis on reconstructing kinematic observables in deep inelastic scattering kinematics with multi-scale neural networks shows that not only are those models effective for real world problems, but the power of the approximation is sufficiently great, that it can outperform reconstruction methods based on physical laws.

I dedicate my dissertation to my family and friends.

ACKNOWLEDGMENTS

I would like to thank Dr. Yuesheng Xu for his support and inspiration as a mathematician. I am grateful to Dr. Markus Diefenthaler for his dedication to furthering our scientific understanding and our riveting conversations. I am also grateful to Dr. Andrii Verbytskyi for teaching me a more extensive understanding of computing and project management. I appreciate the ZEUS Collaboration for allowing the use of ZEUS Monte Carlo simulated samples in this analysis and the Data Preservation efforts [1] in DESY and Max-Planck for Physics that made this analysis possible. I thank R. Aggarwal, C. Fanelli, Y. Furletova, D. Higinbotham, D. Lawrence, S. Menke, A. Vossen, and R. Yoshida for helpful discussions. I was supported in part by National Science Foundation under grant DMS-1912958. I was was also supported by an EIC Center Fellowship at the Thomas Jefferson National Accelerator Facility and a CNF Fellowship with the Center for Nuclear Femtography.

TABLE OF CONTENTS

LIST OF TABLES

Table Page

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

Inference from direct observation is one of the best tools for understanding the world around us. Machine learning, and in particular, deep neural networks, provide an extremely powerful method for making these inferences.

Deep neural networks have successfully been used towards solving a wide range of machine learning problems. Some examples of this include facial recognition, speech recognition, game intelligence, autonomous navigation, and natural language processing. The great success of deep learning [2, 3] and its impact to science, technology and our society have been widely recognized [4, 5, 6, 7, 8, 9, 10, 11].

Compared to the vast development in engineering and applications of deep neural networks, research on the mathematical theory has yet to be completely developed, but is undergoing rapid progress. Many papers on the approximation powers and expressiveness of deep neural networks have been written recently. Here is a brief summary, however, more details can be found in two surveys [12, 13]. Poggio, Mhaskar, Rosasco, Miranda, and Liao [14] proved that deep neural networks approximate a class of functions with special compositional structure exponentially better than shallow networks. Montanelli and Du [15] and Yarotsky [16] estimated the number of parameters needed for deep neural networks to achieve a certain error tolerance in approximating functions in the Koborov space space and differential functions, respectively. Montanelli and Yang [17] achieved error bounds for deep ReLU networks approximation of multivariate functions using the Kolmogorov-Arnold superposition theorem. These three pieces of work indicated that deep neural networks are able to lessen the curse of dimensionality. E and Wang proved that for analytic functions in a low dimension, the convergence rate of the deep neural networks approximation is exponential. Zhou [18] established the universality of deep convolutional neural networks. Daubechies, DeVore, Foucart, Hanin, and Petrova [19] showed that deep neural networks possess greater approximation power than traditional methods of nonlinear approximation such as variable knot splines and n-term approximation from dictionaries. Wang [20] presented a mathematical introduction to generative adversarial nets.

There are four main factors that contribute to the success of deep neural networks. First is the availability of vast amounts of training data. The second is the recent, dramatic, improvements in computing power. The third one is a class of efficient numerical algorithms such as the backpropagation training algorithm via the Stochastic Gradient Decent (SGD), Adaptive Boosting (AdaBoost) algorithms, and the Expectation-Maximization algorithm (EM). The fourth factor is powerful neural network architectures, such as Convolutional Neural Networks (CNN), Long-Short Time Memory (LSTM) networks, Recurrent Neural Networks (RNN), Generative Adversarial Networks (GAN), Deep Belief Networks (DBN), and Residual Networks (ResNet), which provide a superior way of representing data.

GAN's have had particular attention recently, especially with ChatGPT, based on the generative pre-trained transformer, for its detailed responses and vast answers across many domains of knowledge [21].

This work is also focused on fourth factor.

The deep neural networks are learned by solving optimization problems which determine their parameters (weight matrices and bias vectors) that define them with activation functions. The optimization problems are highly nonconvex and have large numbers of parameters. Solving such optimization problems is challenging. Due to having large numbers of parameters, finding a global minimizer is difficult. By employing the stochastic gradient descent method [22, 23] to solve the optimization problems, most likely only local minimizers may be found, since gradient-based optimization starting from random initialization appears to often get stuck in poor solutions [24]. Moreover, convergence of the iteration is very slow. This is a bug computational problem of deep learning.

With an examination of the structure of the current model of deep neural networks, it can be seen that simple information is encoded in the hidden layers that can be used to create a more robust model. The consideration of the hidden layers induces a notion of scale and from it, a multi-scale model can be created. The use of a multi-scale model can lead to smaller models that can provide an equal, if not better, approximation capability.

Furthermore, the current model of neural networks train all parameters needed for the desired deep neural network simultaneously, i.e. in one grade. We can separate the training process into grades by sequentially training hidden layers according to a particular multi-grade training model.

The rest of this work will expand upon these notions and is organized as follows.

In chapter 2, the single-scale, single-grade model is described. The focus is then turned to **ReLU** networks and an observation on the activation domains of the hidden layers. This provides the foundation for the multi-scale neural network model. The multi-scale neural network model is discussed in depth, and it is shown that the class of them form a set of universal approximators dense in the space of continuous functions. It is shown that multi-scale neural networks are resistant to the vanishing gradient problem that effects the single-scale, single-grade neural networks. The multi-scale networks are compared to the multi-grade neural networks. Finally, there is a discussion on common practices for implementing the training regimes to find optimal neural networks given a set of data.

In chapter 3, numerical examples are conducted to compare the different neural network architectures defined in chapter 2.

In chapter 4, an extended analysis is conducted on reconstructing kinematic observables in deep inelastic scattering experiments.

In chapter 5, concluding remarks are provided.

CHAPTER 2

DEEP LEARNING WITH RELU NETWORKS

Many real world problems can be expressed in the context of learning a particular function. Let $s,t \in$. Suppose that there are given N pairs of points $(\mathbf{x}_k, \mathbf{y}_k)$, $k \in_N := \{1,2,...,N\}$, where $\mathbf{x}_k \in^s$ and $\mathbf{y}_k \in^t$. The problem is to learn, from this set of data, a function $\mathbf{f} :^s \to^t$, which maps each $\mathbf{x}_k$ to $\mathbf{y}_k$ and generalizes the intrinsic information embedded in the data. This function $\mathbf{f}$ is called the target function.

Deep neural networks can provide a sufficient expression of the target function because of their robust structure.

A general, fully connected, feed-forward neural network, put simply, is defined to be the consecutive composition of a sequence of both affine transformations and a nonlinear function. These networks are called deep if the sequence is larger.

Let $\Omega \subseteq^s$ be a bounded input domain and ^{t} the output space. For each $i \in_d$, let $m_i \in$. Choose $m_0 = s$.

Select matrices $\mathbf{W}_i \in^{m_i \times m_{i-1}}$ and vectors $\mathbf{b} \in^{m_i}$. $\mathbf{W}_i$ and $\mathbf{b}_i$ are called the weight matrices and bias vectors, respectively.

Select a nonlinear function σ that acts componentwise on an vector input. In other words, for any $p \in$ and $\mathbf{z} = (z_1, z_2, \ldots, z_p)^{\mathrm{T}} \in^p$,

$$\sigma(\mathbf{z}) = (\sigma(z_1), \sigma(z_2), \ldots, \sigma(z_p))^{\mathrm{T}}.$$

This function σ is called the activation function.

A few common activation functions include the **ReLU** function

$$\mathbf{ReLU}(x) := \max(x, 0), \quad x \in, \tag{2.1}$$

the logistic sigmoid function

$$S(x) := \frac{1}{1+e^{-x}}, \quad x \in, \tag{2.2}$$

and the Gaussian function

$$N(x) := e^{-x^2}, \quad x \in . \tag{2.3}$$

There are particular rules on the selection of the activation function that establishes the theoretical basis on which approximation via neural networks is justified. This will be simply expressed later.

For an input $\mathbf{x} \in \Omega$, define the i^{th} hidden layer to be

$$\mathbf{x}^{(i)} = \sigma\left(\mathbf{W}_i \mathbf{x}^{(i-1)} + \mathbf{b}_i\right), \tag{2.4}$$

with $\mathbf{x}^{(0)} = \mathbf{x}$.

Each element in $\mathbf{x}^{(i)}$ is called a node. Each hidden layer has m_i nodes.

Select matrix $\mathbf{W}_o \in^{m_d \times t}$ and $\mathbf{b}_o \in^{t}$. Respectively, $\mathbf{W}_o$ and $\mathbf{b}_o$ are the output weight matrix and the output bias vector. Define

$$\mathbf{y} = \mathbf{W}_o \mathbf{x}^{(d)} + \mathbf{b}_o. \tag{2.5}$$

The selection of the above matrices, vectors, and activation function defines a fully connected, feed-forward neural network that determines a continuous function mapping $\mathbf{x}$ to $\mathbf{y}$ from Ω to ^{t}. This can be conveniently visualized with

$$\mathbf{x} \in \Omega \xrightarrow[\sigma]{\mathbf{W}_1,\mathbf{b}_1} \mathbf{x}^{(1)} \xrightarrow[\sigma]{\mathbf{W}_2,\mathbf{b}_2} \mathbf{x}^{(2)} \to \cdots \to \mathbf{x}^{(d-1)} \xrightarrow[\sigma]{\mathbf{W}_d,\mathbf{b}_d} \mathbf{x}^{(d)} \xrightarrow[\sigma]{\mathbf{W}_o,\mathbf{b}_o} \mathbf{y} \in^{t}. \tag{2.6}$$

The depth of the neural network is defined to be d and the width of hidden layer i is m_i.

To express the neural network in a convenient way, we define the notation for the consecutive composition of functions below.

Definition 2.0.1 (Consecutive composition). *Let $f_1, f_2, f_3, \ldots, f_n$ be a finite sequence of functions such that the range of f_i is contained in the domain of f_{i+1}, $i \in_{n-1}$, the consecutive composition of $\{f_i\}_{i=1}^{n}$ is defined to be the function*

$$\bigodot_{i=1}^{n} f_i := f_n \circ f_{n-1} \circ \cdots f_2 \circ f_1$$

whose domain is the domain of f_1.

Using the notation in definition 2.0.1, equations (2.4) and (2.5) can be rewritten as

$$\mathbf{x}^{(i)} = \left(\bigodot_{l=1}^{i} \sigma\left(\mathbf{W}_l \cdot + \mathbf{b}_l\right)\right)(\mathbf{x}), \quad 1 \leq l \leq d \tag{2.7}$$

and

$$\mathbf{y} = \mathbf{W}_o \left(\bigodot_{l=1}^{d} \sigma\left(\mathbf{W}_l \cdot + \mathbf{b}_l\right)\right)(\mathbf{x}) + \mathbf{b}_o, \quad x \in \Omega \tag{2.8}$$

One indication of the robust and powerful structure of deep neural networks is elucidated in their their classification as universal approximators in the following sense. The universal approximation properties are discussed and proven in reference [25]. Particularly, for any continuous function $f : \Omega \to^t$, for a fixed selection of the depth of the neural networks, and the selection of non-constant activation function, and for any $\varepsilon > 0$, there exists some neural network of the preceding structure, call it ϕ, such that $\rho(f,\phi) < \varepsilon$, where:

$$\rho(f,\phi) = \sup_{\mathbf{x}\in^s} \|f(\mathbf{x}) - \phi(\mathbf{x})\|_{\ell_1}, \tag{2.9}$$

and $\|\cdot\|_{\ell_1}$ is the ℓ^1-norm. That is, for any $\mathbf{x} = (x_1, x_2, ..., x_m)^T \in^s$, $\|\mathbf{x}\|_{\ell_1} = \sum_{i=1}^{s} |x_i|$.

This strong result guarantees the existence of some element in the class of neural networks that is arbitrarily close to any continuous function. For this reason, we say that the set of deep neural networks is dense in the space of continuous functions.

To conclude this initial discussion as a sufficient foundation for the purpose of this work, call this general, fully connected, feed-forward neural network to be a single-scale, single-grade deep neural network.

2.1 THE LEARNING PROBLEM

Now return to the problem of learning a function given a set of data. To approximate the target function $\mathbf{f}$ with N pairs of data points $(\mathbf{x}_k, \mathbf{y}_k)$, $k \in_N$, one can learn a deep neural network

$$\mathbf{y}^*(\mathbf{x}) := \mathbf{y}(\{\mathbf{W}_i^*, \mathbf{b}_i^*\}_{i=1}^{d}; \mathbf{x}), \quad \mathbf{x} \in \Omega \tag{2.10}$$

by solving for optimal parameters $\{\mathbf{W}_i^*, \mathbf{b}_i^*\}_{i=1}^{d}$, the weight matrices and bias vectors, from the optimization problem

$$\min \left\{ \sum_{k=1}^{N} \|\mathbf{y}(\{\mathbf{W}_i, \mathbf{b}_i\}_{i=1}^{d}; \mathbf{x}_k) - \mathbf{y}_k\|_{\ell_2}^2 \quad \middle| \quad \mathbf{W}_i \in^{m_i \times m_{i-1}}, \mathbf{b}_i \in^{m_i}, i \in_d, \mathbf{W}_o \in^{m_i \times t}, \mathbf{b}_o \in^t \right\}, \tag{2.11}$$

where $\|\cdot\|_{\ell_2}$ is the ℓ^2-norm. That is, for any $\mathbf{x} = (x_1, x_2, ..., x_m)^T \in^s$, $\|\mathbf{x}\|_{\ell_2} = \left(\sum_{i=1}^{s} x_i^2\right)^{1/2}$.

The function $\mathbf{y}^*$ is called the best approximation from the set of neural networks with depth d and fixed activation function σ to the function $\mathbf{f}$.

Learning a deep neural network from discrete data is equivalent to finding the optimal weight matrices and bias vectors by solving minimization problem (2.11).

To try to improve the quality of the approximation and computational costs required in solving the minimization problem, by considering information from all hidden layers instead of just the output layer, a notion of scale can be developed to construct a multi-scale deep neural network model. By a further extension, a similarity can be found between the scaled approach and a graduated learning model which focuses on the error of approximation.

These ideas are now discussed in the context of deep neural networks with a **ReLU** activation function.

2.2 RELU NETWORKS

From here on, the activation function will always be the the **ReLU** function. Many results are applicable regardless of the selection of the activation function. In these cases, it will be noted.

It is understood that a **ReLU** network is a piece-wise linear function [12].

To unveil some key properties, an algebraic formulation of a deep **ReLU** network will be introduced, as discussed in reference [26].

First, consider a one-layer **ReLU** network, ϕ.

$$\phi(\mathbf{x}) = \mathbf{W}_o\mathbf{x}^{(1)}(\mathbf{x}) + \mathbf{b}_o, \quad \mathbf{x} \in \Omega \tag{2.12}$$

where

$$\mathbf{x}^{(1)}(\mathbf{x}) = \sigma\left(\mathbf{W}_1\mathbf{x} + b_1\right), \quad x \in \Omega, \tag{2.13}$$

and σ is understood to be the **ReLU** activation function. Note that the m_1 components of $\mathbf{W}_1\mathbf{x} + \mathbf{b}_1$ are linear functions. For convenience, call each component $\ell_j(\mathbf{x})$, for $j = 1, 2, \ldots, m_1$. Therefore,

$$\mathbf{x}^{(1)} = [\sigma(\ell_j(\mathbf{x})) \quad | \quad j = 1, 2, \ldots, m_1]^{\mathrm{T}}. \tag{2.14}$$

By the definition of the **ReLU** function, note that for each $j = 1, 2, \ldots, m_1$,

$$\sigma(\ell_j(\mathbf{x})) = \begin{cases} \ell_j(x), & \ell_j(\mathbf{x}) \geq 0 \\ 0, & \ell_j(\mathbf{x}) < 0. \end{cases} \tag{2.15}$$

A node in $\mathbf{x}^{(1)}$ is said to be activated if $\ell_j(\mathbf{x}) \geq 0$, and it is said to be deactivated if $\ell_j(\mathbf{x}) < 0$. Simply counting will lead to at most 2^{m_1} patterns for the activation status of the nodes in the single layer. The patterns can be described further with a set of of $m_1 \times m_1$ diagonal matrices with diagonal entries either 1 or 0. Particularly, define the set of all activation matrices by

$$\mathscr{D}_{m_1} := \{diag(a_1, a_2, \ldots, a_{m_1}) \quad | \quad a_k \in \{0, 1\}, k \in_{m_1}\}.$$

Define the support of an activation matrix $J \in \mathscr{D}_{m_1}$ by

$$\text{supp } J := \{k \quad | \quad J_{kk} = 1, k \in_{m_1}\}$$

.

It is easily seen that every activation matrix is uniquely determined by its corresponding support. It is convenient to use the set of activation matrices $\mathscr{D}_{m_1}$ as an index set.

Definition 2.2.1 (Activation domains of one layer network)**.** *For a weight matrix* $\mathbf{W}$ *with m rows and a bias vector* $\mathbf{b} \in^m$*, the activation domain of* $\sigma(\mathbf{W}x+\mathbf{b})$ *with respect to a diagonal matrix* $J \in \mathscr{D}_m$ *is*

$$\mathscr{D}_{J,\mathbf{W},\mathbf{b}} := \left\{x \in^{m'} \quad | \quad (\mathbf{W}x+\mathbf{b})_j > 0\, \textit{for } j \in \textit{supp } J \textit{ and } (\mathbf{W}x+\mathbf{b})_j \leq 0\, \textit{for } j \notin \textit{supp } J\right\}$$

Note that the integer m' in Definition 2.2.1 may be selected to be m_0 when it is used for the activation domains of the first layer, or the number of nodes m_i when it is used to define activation domains of the other layers.

In Definition 2.2.1, an activation matrix $J \in \mathscr{D}_{m_1}$ to correspond to an activation pattern of the m_1 components of $\mathbf{W}x+\mathbf{b}$. Therefore, Definition 2.2.1 allows a construction of a partition of the domain Ω that is associated with the piece-wise linear nature of the **ReLU** network ϕ. Specifically,

$$\Omega = \bigcup_{I_1 \in \mathscr{D}_{m_1}} (\mathscr{D}_{I_1,\mathbf{W}_1,\mathbf{b}_1} \cap \Omega) \tag{2.16}$$

With equation (2.16), the single hidden layer can be rewritten

$$\mathbf{x}^{(1)}(x) = I_1(\mathbf{W}_1 x+\mathbf{b}_1), \quad x \in \mathscr{D}_{I_1,\mathbf{W}_1,\mathbf{b}_1}, \quad \text{for } I_1 \in \mathscr{D}_{m_1}. \tag{2.17}$$

It is clear that on each activation domain, $\mathbf{x}^{(1)}$ is a linear function.

For a deep **ReLU** network with d hidden layers, it is necessary to have a sequence of d activation matrices

$$\bar{\mathbf{I}}_d = (I_1, I_2, \ldots, I_d) \in (\mathscr{D}_{m_1}, \mathscr{D}_{m_2}, \ldots, \mathscr{D}_{m_d})$$

where I_k denotes the activation pattern on the k-th hidden layer. Now, here is the definition of activation domains for a multi-layer **ReLU** network.

Definition 2.2.2 (Activation domains of a multi-layer network)**.** *For* $\overline{\mathbf{W}}_d := (\mathbf{W}_1, \mathbf{W}_2, \ldots, \mathbf{W}_d) \in^{m_1 \times m_0} \times^{m_2 \times m_1} \times \cdots \times^{m_d \times m_{d-1}}$*, the weight matrices, and* $\overline{\mathbf{b}}_d = (\mathbf{b}_1, \mathbf{b}_2, \ldots, \mathbf{b}_d) \in^{m_1} \times^{m_2} \times \cdots \times^{m_d}$*, the bias vectors, the activation domain of*

$$\bigodot_{l=1}^{d} \sigma(\mathbf{W}_l \cdot + \mathbf{b}_l)$$

with respect to $\bar{\mathbf{I}}_d = (I_1, I_2, \dots, I_d) \in (\mathscr{D}_{m_1}, \mathscr{D}_{m_2}, \dots, \mathscr{D}_{m_d})$ *is defined recursively by*

$$\mathscr{D}_{\bar{I}_1, \overline{\mathbf{W}}_1, \bar{\mathbf{b}}_1} = \mathscr{D}_{I_1, \mathbf{W}_1, \mathbf{b}_1} \cap \Omega$$

and

$$\mathscr{D}_{\bar{I}_d, \overline{\mathbf{W}}_d, \bar{\mathbf{b}}_d} = \left\{ x \in \mathscr{D}_{\bar{I}_{d-1}, \overline{\mathbf{W}}_{d-1}, \bar{\mathbf{b}}_{d-1}} \quad \Big| \quad \left(\bigodot_{l=1}^{d-1} \sigma\left(\mathbf{W}_l \cdot + \mathbf{b}_l\right) \right)(x) \in \mathscr{D}_{I_d, \mathbf{W}_d, \mathbf{b}_d} \cdot \right\}$$

The following observation is apparent regarding the activation domains.

Theorem 2.2.1. *The sequence of the activation domains are nested, that is, for k* $\in$,

$$\mathscr{D}_{\bar{I}_{k+1}, \overline{\mathbf{W}}_{k+1}, \bar{\mathbf{b}}_{k+1}} \subseteq \mathscr{D}_{\bar{I}_k, \overline{\mathbf{W}}_k, \bar{\mathbf{b}}_k}.$$

Proof. This result follows directly from definition 2.2.2. □

This last result is the critical observation for developing a multi-scale method for using deep **ReLU** networks.

2.3 SCALED LEARNING

Continue to consider the deep **ReLU** network with depth d. The nested nature of the activation domains indicates that each subsequent hidden layer is associated with an activation domain that provides a finer partition of the input domain Ω. This increasing fineness to the partition can be considered to be the scale of the hidden layer. By weighting and summing the information from each hidden layer, we can construct a multi-scale deep neural network. The details of this follow.

Using the notation in definition 2.0.1, the multi-scale model is defined by the function

$$\mathbf{y} = \mathbf{W}_{o,0}\mathbf{x} + \mathbf{b}_{o,0} + \sum_{i=1}^{d} \left(\mathbf{W}_{o,i}\mathbf{x}^{(i)}(\mathbf{x}) + \mathbf{b}_{o,i} \right), \quad x \in \Omega \tag{2.18}$$

where, for $i \in_d$, $\mathbf{x}^{(i)}$ is defined as in equation (2.7), $\mathbf{W}_{o,i} \in^{m_i \times t}$, and $\mathbf{b}_{o,i} \in^t$.

The illustration in figure 1 can help to visualize the multi-scale model.

As previously mentioned, there are results on the natural of single-scale, single-grade deep neural networks as universal approximators in the space of continuous functions. The universality property will be extended to the set of multi-scale networks, whose structure is defined in equation (2.18).

In particular, to guarantee the existence of a function in this class of multi-scale deep neural networks that can approximate any continuous function to arbitrary accuracy, theorem 2.3.1 is now

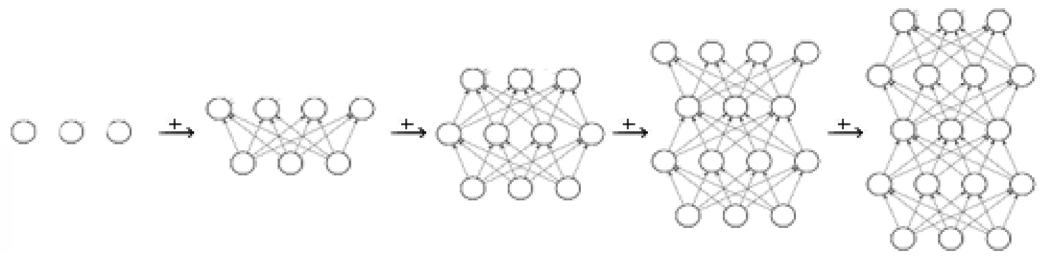


Fig. 1. A visualization of a multi-scale model with five scales as a superposition of five networks.

proven, providing similar universality properties as previously mentioned. Moreover, the theorem also shows that the error necessarily decreases as the depth of the network increases, or, in other words, as the number of terms in the summation in equation (2.18) increases.

Theorem 2.3.1. *With fixed non-zero natural numbers d, s, t, m_1, m_2, ..., m_d, continuous function* $\mathbf{f} : \Omega \subseteq^s \to^t$*, for any* $\varepsilon > 0$*, there exists some* $\mathbf{W}_i \in^{m_i \times m_{i-1}}$*,* $\mathbf{b}_i \in^{m_i}$*,* $\mathbf{W}_{o,i} \in^{m_i \times t}$*,* $\mathbf{b}_{o,i} \in^t$*, for* $i = 0, 1, \ldots, d$*, with* $m_0 = s$*, such that for* $\mathbf{y}$ *defined in Equation* (2.18)*,*

$$\rho(f, \mathbf{y}) < \varepsilon,$$

where ρ *is defined in Equation* (2.9)*.*

Moreover, for $1 \leq k \leq d$*, define*

$$\mathbf{y}_k = \mathbf{W}_{o,0}\mathbf{x} + \mathbf{b}_{o,0} + \sum_{i=1}^{k} \left(\mathbf{W}_{o,i}\mathbf{x}^{(i)}(\mathbf{x}) + \mathbf{b}_{o,i} \right), \quad x \in \Omega,$$

and

$$\varepsilon_k = \rho(\mathbf{f}, \mathbf{y}_k),$$

then the optimal, nontrivial, parameters defining $\mathbf{y}$ *can be selected such that* $\varepsilon_k < \varepsilon_{k-1}$*.*

Proof. We prove the result inductively on the depth d.

For any matrix $\mathbf{W}_{o,0} \in^{m_0 \times t}$ and vector $\mathbf{b}_{o,0} \in^t$, define the continuous function

$$\mathbf{e}_0(\mathbf{x}) := \mathbf{f}(\mathbf{x}) - (\mathbf{W}_{o,0}\mathbf{x} + \mathbf{b}_{o,0}), \quad \mathbf{x} \in \Omega$$

The function $\mathbf{e}_0 : \Omega \subseteq^s \to^t$ is a residual between the target function f and an initial linear approximation.

Due to the universality of single-layer neural networks, as previously mentioned, for any $\varepsilon > 0$ there exists some $\mathbf{W}_1 \in^{m_1 \times m_0}$, $\mathbf{b}_1 \in^{m_1}$, $\mathbf{W}_{o,1} \in^{m_1 \times t}$, $\mathbf{b}_{o,1} \in^t$ such that

$$\rho(\mathbf{e}_0, \mathbf{W}_{o,1}\sigma(\mathbf{W}_1 \cdot + b_1)) + \mathbf{b}_{o,1}) < \varepsilon.$$

Therefore, by definition,

$$\rho(\mathbf{f}, \mathbf{y}_1) < \varepsilon.$$

So, the collection of multi-scale networks with one hidden layer is dense in the space of continuous functions.

Select $\mathbf{W}_1 \in^{m_1 \times m_0}$, $\mathbf{b}_1 \in^{m_1}$, $\mathbf{W}_{o,1} \in^{m_1 \times t}$, $\mathbf{b}_{o,1} \in^t$ to be optimal solutions to the minimization problem:

$$\min\{\rho(\mathbf{f}, \mathbf{y}_1)\},$$

and define $\varepsilon_1 := \rho(\mathbf{f}, \mathbf{y}_1)$.

Continue to define another continuous, residual function, $\mathbf{e}_1 : \Omega \to^t$, by

$$\mathbf{e}_1(\mathbf{x}) = \mathbf{f}(\mathbf{x}) - \mathbf{y}_1(\mathbf{x}). \tag{2.19}$$

Again, due to the universality of single-layer neural networks, there exists some $\mathbf{W}_2 \in^{m_2 \times m_1}$, $\mathbf{b}_2 \in^{m_2}$, $\mathbf{W}_{o,2} \in^{m_2 \times t}$, $\mathbf{b}_{o,2} \in^t$ such that

$$\rho(\mathbf{e}_1, \mathbf{W}_{o,2}\sigma(\mathbf{W}_2 \cdot + b_2)) + \mathbf{b}_{o,2}) < \varepsilon.$$

Therefore, by definition,

$$\rho(\mathbf{f}, \mathbf{y}_2) < \varepsilon.$$

So, the collection of multi-scale networks with two hidden layers is dense in the space of continuous functions.

Again, select $\mathbf{W}_2 \in^{m_2 \times m_1}$, $\mathbf{b}_2 \in^{m_2}$, $\mathbf{W}_{o,2} \in^{m_2 \times t}$, $\mathbf{b}_{o,2} \in^t$ to be optimal solutions to the minimization problem:

$$\min\{\rho(\mathbf{f}, \mathbf{y}_2)\},$$

and define $\varepsilon_2 := \rho(\mathbf{f}, \mathbf{y}_2)$.

Then, excluding any trivial solutions for the previous minimization problem,

$$\varepsilon_2 < \varepsilon_1. \tag{2.20}$$

Now, for fixed $k \in$, assume that the collection of multi-scale neural networks of depth $k-1$ is dense in the space of continuous functions. Therefore, there exists some $\mathbf{W}_i \in^{m_i \times m_{i-1}}$, $\mathbf{b}_i \in^{m_i}$, $\mathbf{W}_{o,i} \in^{m_i \times t}$, $\mathbf{b}_{o,i} \in^{t}$, for $i = 0, 1, \ldots, k-1$, with $m_0 = s$, such that for $\mathbf{y}_{k-1}$,

$$\rho(f, \mathbf{y}_{k-1}) < \varepsilon.$$

Select those parameters to be optimal solutions to the minimization problem:

$$\min \{\rho(\mathbf{f}, \mathbf{y}_k)\},$$

and define $\varepsilon_{k-1} := \rho(\mathbf{f}, \mathbf{y}_{k-1})$.

Continue to define another continuous, residual function, $\mathbf{e}_{k-1} : \Omega \to^{t}$, by

$$\mathbf{e}_{k-1}(\mathbf{x}) = \mathbf{f}(\mathbf{x}) - \mathbf{y}_{k-1}(\mathbf{x}). \tag{2.21}$$

Again, due to the universality of single-layer neural networks, there exists some $\mathbf{W}_k \in^{m_k \times m_{k-1}}$, $\mathbf{b}_k \in^{m_k}$, $\mathbf{W}_{o,k} \in^{m_k \times t}$, $\mathbf{b}_{o,k} \in^{t}$ such that

$$\rho(\mathbf{e}_{k-1}, \mathbf{W}_{o,k}\sigma(\mathbf{W}_k \cdot + b_k)) + \mathbf{b}_{o,k}) < \varepsilon.$$

Therefore, by definition,

$$\rho(\mathbf{f}, \mathbf{y}_k) < \varepsilon.$$

So, the collection of multi-scale networks with k hidden layers is dense in the space of continuous functions.

Again, select $\mathbf{W}_k \in^{m_k \times m_{k-1}}$, $\mathbf{b}_k \in^{m_k}$, $\mathbf{W}_{o,k} \in^{m_k \times t}$, $\mathbf{b}_{o,k} \in^{t}$ to be optimal solutions to the minimization problem:

$$\min \{\rho(\mathbf{f}, \mathbf{y}_k)\},$$

and define $\varepsilon_k := \rho(\mathbf{f}, \mathbf{y}_k)$.

Then, again, excluding any trivial solutions for the previous minimization problem,

$$\varepsilon_k < \varepsilon_{k-1}. \tag{2.22}$$

This concludes the proof that the collection of multi-scale neural networks is dense in the space of continuous functions and that increasing the depth necessarily reduces the approximation error.

□

The previous result is not limited to only **ReLU** activation functions and justifies the use of multi-scale neural networks, in the sense that they can universally approximate any continuous function. Furthermore, the nature of their structure indicates that the approximation error strictly decreases as the number of hidden layers gets larger.

The fact that the increasing the number layers reduces the approximation error leads one to believe that a very deep network can be used to solve many tasks. Indeed, many nontrivial problems have greatly benefited from very deep models [27, 28, 29, 30, 31, 32, 33]. An issue that arises with training single-scale, single-grade neural networks is the problem of vanishing [34, 35]. The problem has been addressed in through techniques like normalized initialization [35, 29, 36, 37], and intermediate normalization layers [38], which enable deep networks start converge for stochastic gradient descent algorithm with backpropagation [39]. The structure of the multi-scale model is such that this problem is essentially avoided all together. Some more details on this are provided at the end of section 2.5.

Now, recognizing that the multi-scale network is a superposition of the hidden layers, we can consider a graduated learning model, where subsequent hidden layers are trained consecutively. This is described in the next section.

2.4 GRADUATED LEARNING

Learning a deep neural network from discrete data reduces to the task of finding the optimal weight matrices and bias vectors by solving a minimization problem. The single-grade learning model learns the parameters of all layers together in one single grade.

After a deep neural network is learned, if its accuracy is not satisfactory, then the training regime needs to be repeated with more layers. This is not computationally efficient. To allow updating the learned neural network to form a new one without training a complete new neural network, the multi-grade learning model proposed in reference [40] addresses these issues. Here, some specifics of the model are discussed.

In the multi-grade learning model, each grade is composed of training one set of parameters that, when superposed, form the entire multi-grade model.

For convenience, generally express a neural network with depth n in the following way:

$$\mathbf{N}_n(\mathbf{x}) = \mathbf{W}_o \left(\bigodot_{l=1}^{n} \sigma\left(\mathbf{W}_l \cdot + \mathbf{b}_l\right) \right)(\mathbf{x}) + \mathbf{b}_o, \quad x \in \Omega \tag{2.23}$$

with weight matrices and bias vectors defined precisely as before.

To describe the multi-grade model, organize the training process into g grades. Particularly, choose $n_j \in, j \in_g$, so that $\sum_{j=1}^{g} n_j = n$.

Begin with grade 1. The goal is to learn the neural network $\mathbf{N}_{n_1}$ that approximates the target function $\mathbf{f} : \Omega \subseteq^s \to^t$.

Define the error function of grade 1 by

$$\mathbf{e}_1(\mathbf{x}) = \mathbf{f}(\mathbf{x}) - \mathbf{N}_{k_1}(\mathbf{x}) \tag{2.24}$$

where the parameters $\mathbf{W}_j \in^{m_j \times m_{j-1}}$, $\mathbf{b}_j \in^{m_j}$, $\mathbf{W}_o \in^{m_{k_1} \times t}$, $\mathbf{b}_o \in^t$, for $j = 1, 2, \ldots, k_1$ are to be learned from solving the optimization problem:

$$\min \left\{ \|\mathbf{e}_1(\mathbf{x})\|_{\ell_2}^2 \right\} \tag{2.25}$$

Let $\mathbf{N}_{k_1}^*$ defined by these optimal parameters and define

$$\mathbf{f}_1(\mathbf{x}) = \mathbf{N}_{k_1}^*(\mathbf{x}),$$

which is information about $\mathbf{f}$ learned in grade 1. Define the optimal error of grade 1 by

$$\mathbf{e}_1^*(\mathbf{x}) = \mathbf{f}(\mathbf{x}) - \mathbf{f}_1(\mathbf{x}) \tag{2.26}$$

Usually the magnitude of the optimal error of grade 1 is not sufficiently small, so training will continue to the next grade.

In grade 2, another shallow neural network will be learned on top of vf_1 by approximating the optimal error $\mathbf{e}_1^*$ of grade 1.

Define the error function of grade 2 as

$$\mathbf{e}_2(\mathbf{x}) = \mathbf{f}(\mathbf{x}) - \mathbf{N}_{k_2}(\mathbf{x}) \tag{2.27}$$

where the parameters $\mathbf{W}_j \in^{m_j \times m_{j-1}}$, $\mathbf{b}_j \in^{m_j}$, $\mathbf{W}_o \in^{m_{k_1} \times t}$, $\mathbf{b}_o \in^t$, for $j = 1, 2, \ldots, k_2$ are to be learned from solving the optimization problem:

$$\min \left\{ \|\mathbf{e}_2(\mathbf{x})\|_{\ell_2}^2 \right\} \tag{2.28}$$

Let $\mathbf{N}_{k_2}^*$ defined by these optimal parameters and define

$$\mathbf{f}_2(\mathbf{x}) = (\mathbf{N}_{k_2}^* \circ \mathbf{N}_{k_1}^*)(\mathbf{x}),$$

which is the new network stacked on top of the previous one. Define the optimal error of grade 2 by

$$\mathbf{e}_2^*(\mathbf{x}) = \mathbf{e}_1^*(\mathbf{x}) - \mathbf{f}_2(\mathbf{x}) \tag{2.29}$$

Now, suppose that the neural networks $\mathbf{N}_{k_j}^*$, for $j \in$, have been learned and the optimal error for grade i is given by $\mathbf{e}_i^*$. The error function for grade $i+1$ is

$$\mathbf{e}_{i+1}^*(\mathbf{x}) = \mathbf{e}_i^*(\mathbf{x}) - (\mathbf{N}_{k_{i+1}}^* \circ \mathbf{N}_{k_i}^* \circ \cdots \circ \mathbf{N}_{k_1}^*)(\mathbf{x}) \tag{2.30}$$

where $\mathbf{N}^*_{k_{i+1}}$ is to be learned in grade $i+1$ similarly to above by solving the optimization problem:

$$\min\left\{\|\mathbf{e}_{i+1}(\mathbf{x})\|^2_{\ell_2}\right\}. \tag{2.31}$$

Define

$$\mathbf{f}_{i+1}(\mathbf{x}) = (\mathbf{N}^*_{k_{i+1}} \circ \mathbf{N}^*_{k_i} \circ \cdots \circ \mathbf{N}^*_{k_1})(\mathbf{x}). \tag{2.32}$$

If this process is repeated for all g grades, then finally, the g grade learning model generates the neural network

$$\bar{\mathbf{f}}_g = \sum_{i=1}^{g} \mathbf{f}_i \tag{2.33}$$

Unlike the single-scale, single-grade neural network, the neural network $\bar{\mathbf{f}}_g$ defined by in equation (2.33) learned by the g grade model is the sum total of the information learned from all the grades. Unlike the multi-scale neural network, the multi-grade neural network is equipped with a directly approach to approximating the error of approximations with more shallow neural networks. In each grade, the system learns based on the knowledge gained from learning of the previous grades. Mathematically, the multi-grade neural network is the superposition of the networks in each of the g grades, and each of these is a shallow network.

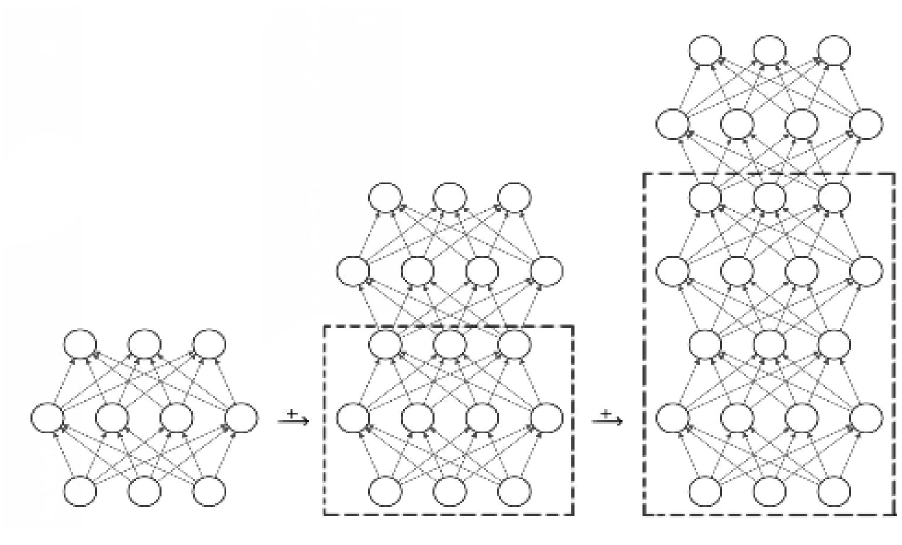


Fig. 2. A visualization of a multi-grade model with three as a superposition of three networks.

In general, the multi-grade neural network has a stairs-shape. In Figure 2, there is an illustration of a three grade learning model, the sum of three networks: the network (left) learned from grade 1 plus the network (center) learned from grade 2, which is stacked on the top of the network learned in grade 1, and plus the network (right) learned from grade 3, which is stacked on the top of the network learned in grade 2. The sections enclosed by dotted lines exact copies of the networks learned from the previous grades and remain unchanged in training of a new grade.

2.5 IMPLEMENTATION

Now, three distinct neural network architectures have been introduced with corresponding learning regimes. What follows in the remainder of this chapter is a brief discussion on the implementation of the training regimes, or, in other words, solving for the optimal solutions of the various minimization problems proposed above.

Recall the learning problem. Given a set of data, approximate the target function $\mathbf{f}:\Omega\subseteq^{s}\to^{t}$ with N pairs of data points $(\mathbf{x}_k,\mathbf{y}_k)$, $k\in_N$.

Call the desired approximator the function $\phi:\Omega\to^{t}$, which is either a single-scale, single grade neural network, a multi-scale neural network, or a multi-grade neural network.

Regardless of the selection of the type of network, each is determined by the elements of the weight matrices and the bias vectors. For simplicity, let $\omega\in^{p}$ be the collection of all these elements, where p is the total number of them. So, for any $\mathbf{x}\in\Omega$, express the network by

$$\phi(\mathbf{x})=\phi_\omega(\mathbf{x}).$$

Choosing an optimal function from this class entails finding the collection of parameters ω in ^{p} that minimises an empirical error over the given data set:

$$\mathscr{L}(\omega):=\frac{1}{N}\sum_{i=1}^{N}\ell(\omega,\mathbf{x}_i,\mathbf{y}_i). \tag{2.34}$$

where ℓ is some loss function measuring the discrepancy between the observed target value $\mathbf{y}_i$ and the predicted value $\phi_\omega(\mathbf{x})$, for $i\in_N$.

This summation in equation (2.34) is sometimes called the fidelity term, as it measures the discrepancy between the data and model predictions. A commonly used fidelity term is the mean square error, with:

$$\ell(\omega,\mathbf{x}_i,\mathbf{y}_i)=\|\mathbf{y}_i-\phi_\omega(\mathbf{x}_i)\|_{\ell_2}^2. \tag{2.35}$$

Another is the the mean square logarithmic error:

$$\ell(\omega,\mathbf{x}_i,\mathbf{y}_i)=\|\log(\mathbf{y}_i)-\log(\phi_\omega(\mathbf{x}_i))\|_{\ell_2}^2. \tag{2.36}$$

Due to the universality of neural networks, there is a model that can achieve zero empirical error. However, in the presence of noise this means the model can overfit to the data sample and loose its generalizability [41].

This problem can be addressed by adding a regularization term to the optimization problem as a penalty for certain irregular behaviour. Common regularization terms include the ℓ_2-norm, used to limit the size of the parameters, and the ℓ_1-norm, which can induce sparse solutions.

The Theorem 1.3 in [42] and Proposition 27 of [43] provide an evidence that minimising the ℓ^1 norm provides sparse optimal solutions with a minimal number of nonzero elements. Well-constructed models should be able to generalize the information from one given sample to any possible event. Thus, regularization with the ℓ_1 norm produces a model determined by a minimal number of parameters so that the optimal solution does not fit completely to the training set and loose its generalizability.

Therefore, the determination of the optimal neural network model consists of minimizing this final loss function, the sum of the sample fidelity term and a weighted regularization term:

$$\min_{\omega \in^p} \frac{1}{N} \sum_{i=1}^{N} \ell(\omega, \mathbf{x}_i, \mathbf{y}_i) + R \cdot ||\omega||_1, \tag{2.37}$$

where ℓ is a pre-selected loss function and R is a hyperparameter to determine the magnitude of the regularization.

Expression (2.37) can be minimised using stochastic gradient methods on batches of the data sample [44, 23, 22].

The training can accelerated using classical momentum methods [45]. In particular, randomly select an initial set of parameters ω^0.

Select a sequence of step sizes, or learning rates, L^k that diminish to zero.

Randomly selecting a batch of data with indices $I \subset \{1, ..., N\}$.

Choose a momentum parameter μ.

By defining:

$$v^{k+1} = \mu v^k - L^k \nabla_\omega \mathscr{L}_I(\omega^k), \tag{2.38}$$

$$\omega^{k+1} = \omega^k + v^{k+1}. \tag{2.39}$$

where

$$\mathscr{L}_I(\omega) := \frac{1}{|I|} \sum_{i \in I} \ell(\omega, \mathbf{x}_i, \mathbf{y}_i),$$

Then the sequence ω^k converges to a set of parameters defining the optimal neural network with the minimal generalisation error, in the sense described here.

There are further extensions of the stochastic gradient methods described above. Particularly, the ADAM method [46], which will be used for some computational examples that follow.

To conclude this section, some details will be provided on how the multi-scale method can overcome the previously mentioned vanishing gradient problem.

Begin by considering the single-scale, single-grade neural network in equation (2.8).

Consider the task of finding the optimal solution while minimizing equation (2.34), with the mean square error for the loss function, using the stochastic gradient descent algorithm with no momentum. To do so, the gradient of the loss function needs to be computed. Observe the value of the partial derivatives for the weight matrix of the first hidden layer has in it a product of the partial derivatives every subsequent layer by the chain rule.

Here is where the vanishing gradient problem arises - when the derivatives of the weights in the later hidden layers approach zero, the weights in the first hidden layer essentially stop updating. This will either cause the stochastic gradient descent algorithm not to converge, or to converge to some suboptimal solution.

By contrast, consider the multi-scale neural network in equation (2.18).

Again, consider the the value of the partial derivatives for the weight matrix. Notice that when it is computed, because of the summation, there is a term that is completely independent of the partial derivatives of the weight matrices from every subsequent layer by the chain rule. This effectively frees the weights in all layers to be updated without influence from diminishing gradient values of other layers. This allows a very deep networks to have a convergent training regime.

These remarks are formalized in the following theorem. First, two lemmas are expressed.

This first is in regards to single-scale, single-grade neural networks.

Lemma 2.5.1. *In a single-scale, single-grade neural network, the partial derivatives of the loss function with respect to parameters from any hidden layer include a product of partial derivatives with respect to each subsequent layer.*

Proof. The single-scale, single-grade neural network is defined by

$$\mathbf{y} = \mathbf{W}_o \mathbf{x}^{(d)} + \mathbf{b}_o,$$

where

$$\mathbf{x}^{(i)} = \sigma\left(\mathbf{W}_i \mathbf{x}^{(i-1)} + \mathbf{b}_i\right), \quad i \in_d,$$

is the i-th hidden layer, with $\mathbf{x}^{(0)} = \mathbf{x}$.

Take the loss function to be minimized to be

$$\mathscr{L} = \frac{1}{N}\sum_{i=1}^{N} \|\mathbf{y}_i - \mathbf{y}(\mathbf{x}_i)\|_{\ell_2}^2.$$

Consider its partial derivatives with respect to the weights of the i-th hidden layer, $\mathbf{W}_i$.

$$\begin{aligned}\frac{\partial \mathscr{L}}{\partial \mathbf{W}_i} &= \frac{\partial \mathscr{L}}{\partial \mathbf{y}} \frac{\partial \mathbf{y}}{\partial \mathbf{W}_i} \\ &= \frac{\partial \mathscr{L}}{\partial \mathbf{y}} \frac{\partial \mathbf{y}}{\partial \mathbf{x}^{(d)}} \frac{\partial \mathbf{x}^{(d)}}{\partial \mathbf{W}_i} \\ &= \frac{\partial \mathscr{L}}{\partial \mathbf{y}} \frac{\partial \mathbf{y}}{\partial \mathbf{x}^{(d)}} \frac{\partial \mathbf{x}^{(d)}}{\partial \mathbf{x}^{(d-1)}} \frac{\partial \mathbf{x}^{(d-1)}}{\partial \mathbf{W}_i}.\end{aligned}$$

Note that for any $j \in_d$,

$$\frac{\partial \mathbf{x}^{(j)}}{\partial \mathbf{W}_i} = \frac{\partial \mathbf{x}^{(j)}}{\partial \mathbf{x}^{(j-1)}} \frac{\partial \mathbf{x}^{(j-1)}}{\partial \mathbf{W}_i}.$$

So, by induction,

$$\frac{\partial \mathscr{L}}{\partial \mathbf{W}_i} = \frac{\partial \mathscr{L}}{\partial \mathbf{y}} \frac{\partial \mathbf{y}}{\partial \mathbf{x}^{(d)}} \left(\prod_{j=i+1}^{d} \frac{\partial \mathbf{x}^{(d+i-j+1)}}{\partial \mathbf{x}^{(d+i-j)}} \right) \frac{\partial \mathbf{x}^{(i)}}{\partial \mathbf{W}_i}$$

□

The second lemma is in regards to multi-scale neural networks.

Lemma 2.5.2. *In a multi-scale neural network, the partial derivatives of the loss function with respect to parameters from any hidden layer have one term that is independent from partial derivatives with respect to any subsequent layer.*

Proof. The multi-scale network to be found is given by

$$\mathbf{y} = \mathbf{W}_{o,0}\mathbf{x} + \mathbf{b}_{o,0} + \sum_{i=1}^{d} \left(\mathbf{W}_{o,i}\mathbf{x}^{(i)}(\mathbf{x}) + \mathbf{b}_{o,i} \right).$$

where where

$$\mathbf{x}^{(i)} = \sigma\left(\mathbf{W}_i \mathbf{x}^{(i-1)} + \mathbf{b}_i\right), \quad i \in_d,$$

is the i-th hidden layer, with $\mathbf{x}^{(0)} = \mathbf{x}$.

Take the loss function to be minimized to be

$$\mathscr{L} = \frac{1}{N} \sum_{i=1}^{N} \|\mathbf{y}_i - \mathbf{y}(\mathbf{x}_i)\|_{\ell_2}^2.$$

Consider the partial derivatives with respect to the weights of the i-th hidden layer, $\mathbf{W}_i$.

$$\begin{aligned}\frac{\partial \mathscr{L}}{\partial \mathbf{W}_i} &= \frac{\partial \mathscr{L}}{\partial \mathbf{y}} \frac{\partial \mathbf{y}}{\partial \mathbf{W}_i} \\ &= \frac{\partial \mathscr{L}}{\partial \mathbf{y}} \sum_{j=1}^{d} \left(\mathbf{W}_{o,j} \frac{\partial \mathbf{x}^{(j)}}{\partial \mathbf{W}_i} \right)\end{aligned}$$

Since, for $i > j$,

$$\frac{\partial \mathbf{x}^{(i)}}{\partial \mathbf{W}_i} = 0,$$

then,

$$\frac{\partial \mathscr{L}}{\partial \mathbf{W}_i} = \sum_{j=i}^{d} \left(\mathbf{W}_{o,j} \frac{\partial \mathbf{x}^{(j)}}{\partial \mathbf{W}_i} \right).$$

Using the result from lemma 2.5.1,

$$\frac{\partial \mathscr{L}}{\partial \mathbf{W}_i} = \sum_{j=i}^{d} \left(\mathbf{W}_{o,j} \left(\prod_{k=i+1}^{j} \frac{\partial \mathbf{x}^{(j+i-k+1)}}{\partial \mathbf{x}^{(j+i-k)}} \right) \frac{\partial \mathbf{x}^{(j)}}{\partial \mathbf{W}_i} \right).$$

For convenience, when the lower limit on the product is larger than the upper limit, set the product equal to one.

Notice that the first term in the above sum, when $j = i$, is

$$\mathbf{W}_{o,i} \frac{\partial \mathbf{x}^{(i)}}{\partial \mathbf{W}_i}.$$

$\square$

Finally, the main theorem on how the multi-scale is suited to overcome the vanishing gradient problem for single-scale, single-grade neural networks is presented.

Theorem 2.5.1. *For a neural network of depth d, and for all $j \in_d$, if there exists some $\xi \in (0,1)$ such that*

$$0 \leq \frac{\partial \mathbf{x}^{(j)}}{\partial \mathbf{x}^{(j-1)}} < \xi \quad \text{and} \quad 0 \leq \frac{\partial \mathbf{y}}{\partial \mathbf{x}^{(d)}} < \xi,$$

or

$$-\xi < \frac{\partial \mathbf{x}^{(j)}}{\partial \mathbf{x}^{(j-1)}} \leq 0 \quad \text{and} \quad -\xi < \frac{\partial \mathbf{y}}{\partial \mathbf{x}^{(d)}} \leq 0,$$

then, for $i \in$,

1. *for a single-scale, single-grade neural network,*

$$\left| \frac{\partial \mathscr{L}}{\partial \mathbf{W}_i} \right| < \left| \frac{\partial \mathscr{L}}{\partial \mathbf{y}} \right| |\xi|^{d-i} \left| \frac{\partial \mathbf{x}^{(i)}}{\partial \mathbf{W}_i} \right| \quad \text{and} \quad \lim_{d \to \infty} \left| \frac{\partial \mathscr{L}}{\partial \mathbf{W}_i} \right| = 0,$$

and

2. *for a multi-scale neural network,*

$$\left| \frac{\partial \mathscr{L}}{\partial \mathbf{W}_i} \right| > \left| \frac{\partial \mathscr{L}}{\partial \mathbf{W}_i} \right| |\xi| \left| \frac{\partial \mathbf{x}^{(i)}}{\partial \mathbf{W}_i} \right|.$$

Proof. First, consider a single-scale, single-grade neural network.

Using lemma 2.5.1,

$$\begin{aligned}\left|\frac{\partial\mathscr{L}}{\partial\mathbf{W}_i}\right| &= \left|\frac{\partial\mathscr{L}}{\partial\mathbf{y}}\frac{\partial\mathbf{y}}{\partial\mathbf{x}^{(d)}}\left(\prod_{j=i+1}^{d}\frac{\partial\mathbf{x}^{(d+i-j+1)}}{\partial\mathbf{x}^{(d+i-j)}}\right)\frac{\partial\mathbf{x}^{(i)}}{\partial\mathbf{W}_i}\right|\\ &< \left|\frac{\partial\mathscr{L}}{\partial\mathbf{y}}\right|\xi^{d-i}\left|\frac{\partial\mathbf{x}^{(i)}}{\partial\mathbf{W}_i}\right|.\end{aligned}$$

Letting the depth of the network approach infinity,

$$\lim_{d\to\infty}\left|\frac{\partial\mathscr{L}}{\partial\mathbf{W}_i}\right| = 0.$$

Now, consider a multi-scale neural network.

Using lemma 2.5.2,

$$\begin{aligned}\left|\frac{\partial\mathscr{L}}{\partial\mathbf{W}_i}\right| &= \left|\frac{\partial\mathscr{L}}{\partial\mathbf{y}}\sum_{j=i}^{d}\left(\mathbf{W}_{o,j}\left(\prod_{k=i+1}^{j}\frac{\partial\mathbf{x}^{(j+i-k+1)}}{\partial\mathbf{x}^{(j+i-k)}}\right)\frac{\partial\mathbf{x}^{(j)}}{\partial\mathbf{W}_i}\right)\right|\\ &= \left|\frac{\partial\mathscr{L}}{\partial\mathbf{y}}\right|\sum_{j=i}^{d}\left(|\mathbf{W}_{o,j}|\prod_{k=i+1}^{j}\left|\frac{\partial\mathbf{x}^{(j+i-k+1)}}{\partial\mathbf{x}^{(j+i-k)}}\right|\left|\frac{\partial\mathbf{x}^{(j)}}{\partial\mathbf{W}_i}\right|\right)\\ &\geq \left|\frac{\partial\mathscr{L}}{\partial\mathbf{y}}\right||\mathbf{W}_{o,i}|\left|\frac{\partial\mathbf{x}^{(i)}}{\partial\mathbf{W}_i}\right|.\end{aligned}$$

By applying the assumed bounds in the theorem,

$$\left|\frac{\partial\mathscr{L}}{\partial\mathbf{W}_i}\right| > \left|\frac{\partial\mathscr{L}}{\partial\mathbf{y}}\right||\xi|\left|\frac{\partial\mathbf{x}^{(i)}}{\partial\mathbf{W}_i}\right|.$$

□

Including the assumptions on the sign and magnitude of the derivatives of the loss function, the consequences of this theorem are that the single-scale, single-grade neural network will succumb to the vanishing gradient problem as the depth gets larger. Meanwhile, the multi-scale neural network has a lower bound on the partial derivatives involved in the gradient based descent methods that allows it to be resistant to the same vanishing gradient problem.

It will be seen that the multi-scale neural network outperforms the single-scale, single-grade neural network in terms of minimizing the loss function. This is in part due to the above theorem, that the multi-scale neural network is resistant to the vanishing gradient problem.

Now, all three models described previously can provide optimal solutions to the minimization problems given a set of data. Even with theoretical comparisons of the methods, there is still the task of practically implementing the optimization methods with the appropriate selection of the training regime.

The following chapter provides such an implementation for three examples with either periodic or non-smooth behaviour.

In practice, there are different things to consider, and so here, two additional models are provided in addition to the standard, single-scale, single-grade deep neural networks as a tool to add additional robustness to desired approximators and/or a reduction in computational complexity.

CHAPTER 3

NUMERICAL EXAMPLES

Here, three numerical examples are provided as a proof-of-concept to compare the robustness of the single-scale, single-grade network, the multi-scale network, and the multi-grade model. Two periodic functions and one nonsmooth function are considered. These cases are considered both with noise and without noise.

All three of the experiments reported in this chapter are performed with Python on an Intel(R) Xeon Gold 6310 CPU at 2.1GHz with two Nvidia Tesla V100 GPU to accelerate the computing.

Below is a description of the experimental data set for a given function f dependent on the example. For $N = 8000$, $\{(x_n, y_n)\}_{n=1}^{N} \subset [a,b]\times$ is the data set, where x_n's are equally spaced on $[a,b]$, and given x_n, the corresponding y_n is computed by $y_n = f(x_n) + e_n$. For the noise free case, $e_n = 0$, and for the noisy case, the e_n's are independent and identically distributed Gaussian random variables with mean 0 and standard deviation 0.05.

Of this data set, 80% is randomly selected for the training set and the remaining 20% is selected for the testing set.

The loss function selected for training the models is the mean square error. In these examples, no regularization term is incorporated into the loss function.

All models were trained using the ADAM optimization method with a learning rate decay of 10^{-2}.

The architectures of the models are described here. Note that the activation function for every layer is the **ReLU** function.

- Single-scale, single-grade (SSG):

  The architecture can be described by the following schematic.

  $$[1] \to [256] \to [256] \to [128] \to [128] \to [64] \to [64] \to [32] \to [32] \to [1] \tag{3.1}$$

  Here, for any $n \in$, $[n]$ indicates a fully-connected layer with n neurons.

- Multi-scale (MS):

There are four multi-scale models considered, corresponding to the following networks

MS-2: $[1] \to [256] \to [256] \to [1]$

MS-4: $[1] \to [256] \to [256] \to [128] \to [128] \to [1]$

MS-6: $[1] \to [256] \to [256] \to [128] \to [128] \to [64] \to [64] \to [1]$

MS-8: $[1] \to [256] \to [256] \to [128] \to [128] \to [64] \to [64] \to [32] \to [32] \to [1]$

Note that the output of each hidden layer is weighted and summed to the produce the final output. All four of the above multi-scale networks are trained independently.

- Multi-grade (MG):

  The multi-grade model has three grades.

  Grade 1: $[1] \to [256] \to [256] \to [1]$

  Grade 2: $[1] \to [256]_F \to [256]_F \to [128] \to [128] \to [64] \to [1]$

  Grade 3: $[1] \to [256]_F \to [256]_F \to [128]_F \to [128]_F \to [64]_F \to [64] \to [32] \to [32] \to [1]$

  Here, similarly, for any $n \in$, $[n]_F$ indicates $[n]$ with all parameters fixed during training.

3.1 EXAMPLE 1

Consider the task of approximating the periodic function

$$f(x) = \sin(24\pi x), \quad x \in [0,1] \tag{3.2}$$

For training all models, 300 epochs are used. The single-scale, single-grade network and the multi-scale networks use a batch size of 16 and an initial learning rate of 10^-4. The number of epochs, batch size, and initial learning rate per grade for the multi-grade model are described in Table 2 for the noise-free case and Table 5 for the noisy case.

The training data for the noise-free case are plotted in Figure 3.

Numerical results for training for this noise-free case are listed in Tables 3, 1, and 2. A corresponding figure is plotted in Figure 4.

The prediction from each method for the noise-free case is plotted in Figure 7. A plot of output from each scale of the multi-scale method with 8 scales is given in Figure 5. A plot of the output from each grade of the multi-grade method with 3 grades is given in Figure 6.

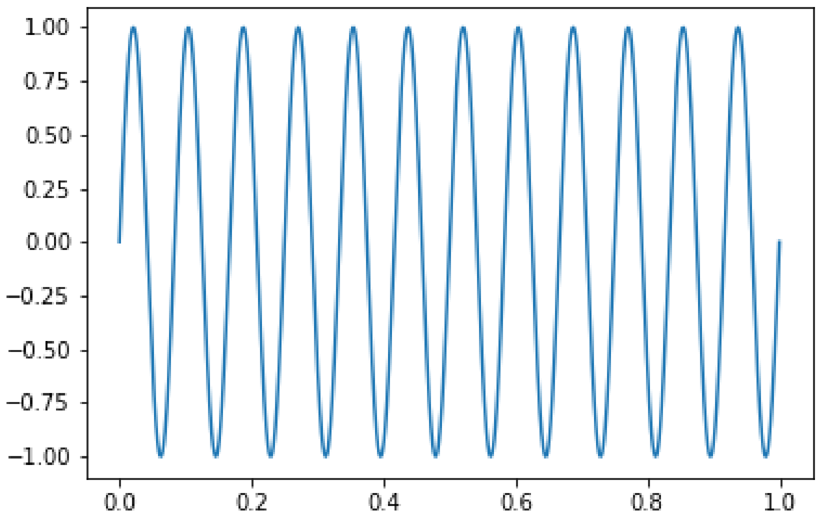


Fig. 3. The example 1 (noise free) target data set.

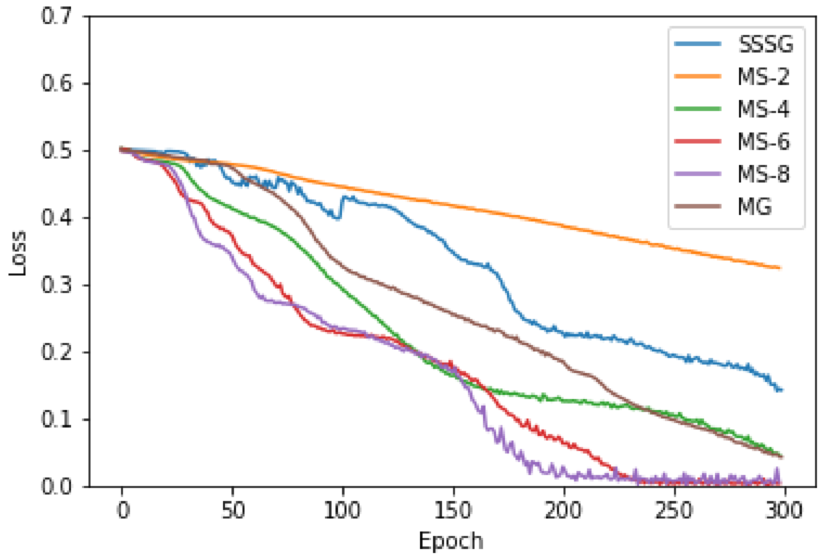


Fig. 4. For example 1 (noise free), the loss curves during training for the single-scale, single grade method, the multi-scale method with two, four, six, and eight scales, and the multi-grade method with 3 grades.

TABLE 1

For example 1 (noise free), training time and accuracy multi-scale model with varying scales.

Scale	Training Time (seconds)	MSE
2	289.9385	0.3238
4	340.2780	0.0466
6	384.3652	0.0037
8	418.5224	0.0065

TABLE 2

For example 1 (noise free), the training time and accuracy for each grade of the Multi-grade model.

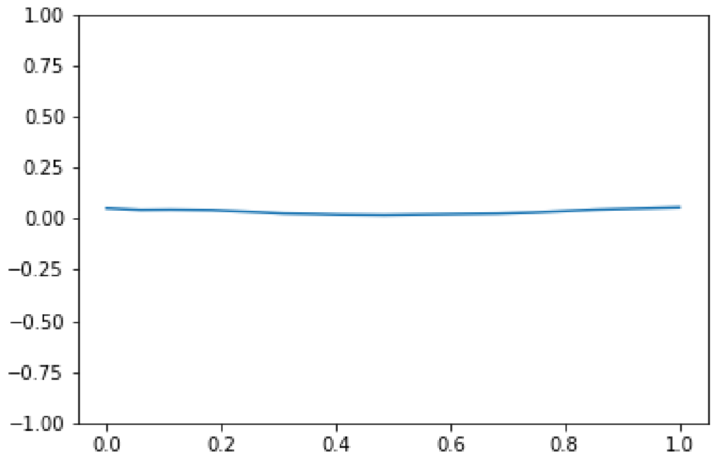
Grade	Learning Rate	Batch Size	Epochs	Training Time (seconds)	MSE
1	10^{-4}	16	40	39.7909	0.4829
2	10^{-4}	16	160	150.3363	0.1863
3	10^{-4}	4	100	311.1016	0.0429

TABLE 3

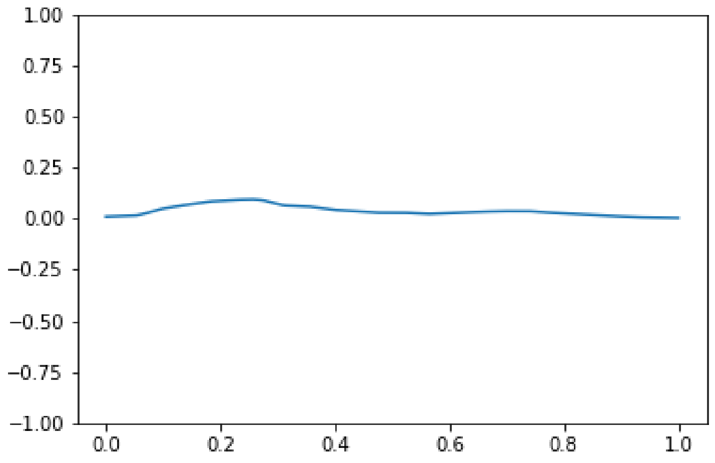
For example 1 (noise free), a comparison of training time and accuracy for each model.

Method	Training Time (seconds)	MSE
Single-scale, single-grade	371.4200	0.1423
Multi-scale	418.5224	0.0065
Multi-grade	501.2288	0.0429

A



B



C

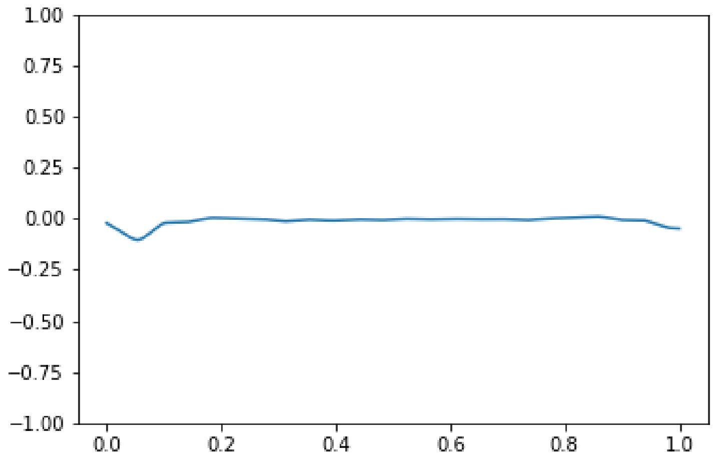
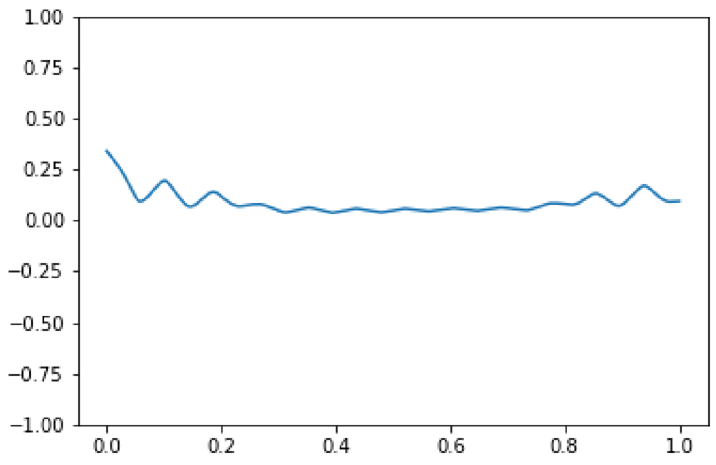
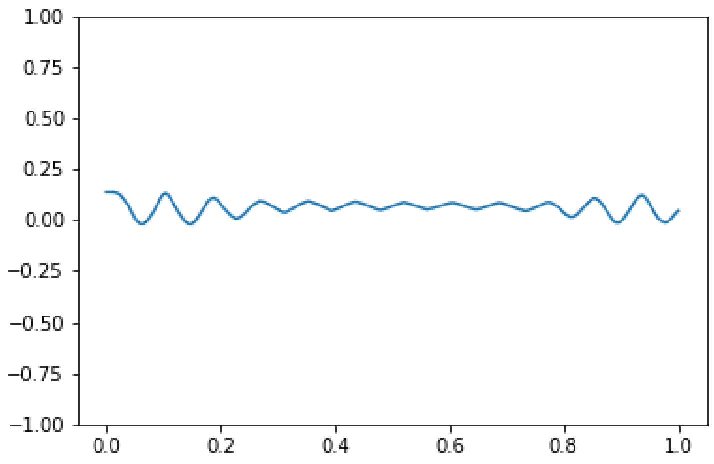


Fig. 5. For example 1 (noise free), a comparison of each scale in the multi-scale method, where each figure labeled A through H corresponds to scales one through eight, respectively.

D



E



F

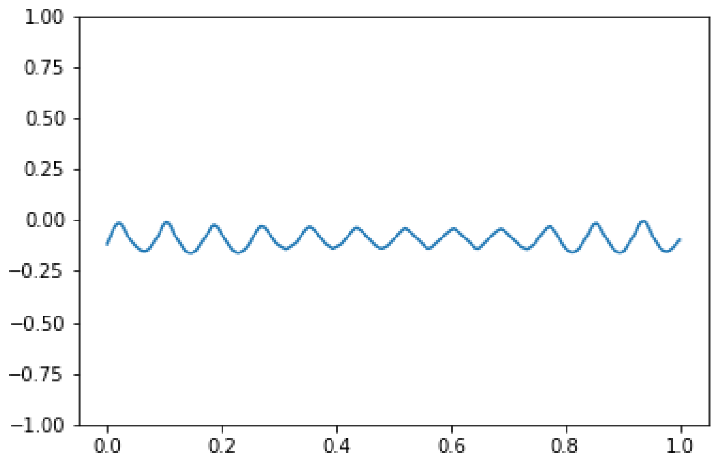
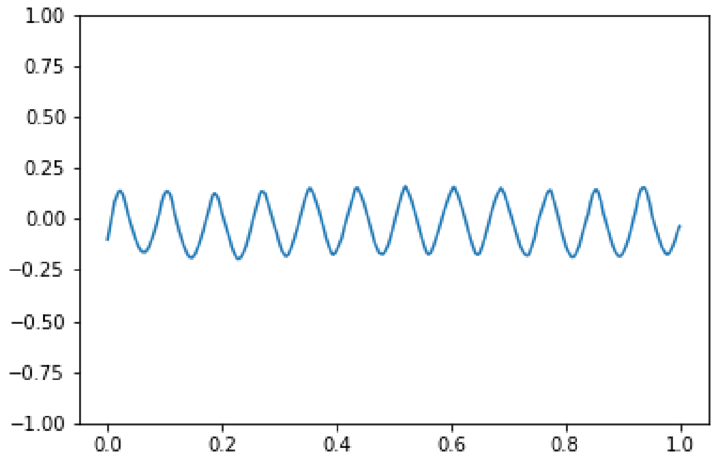


Fig. 5. Continued.

G



H

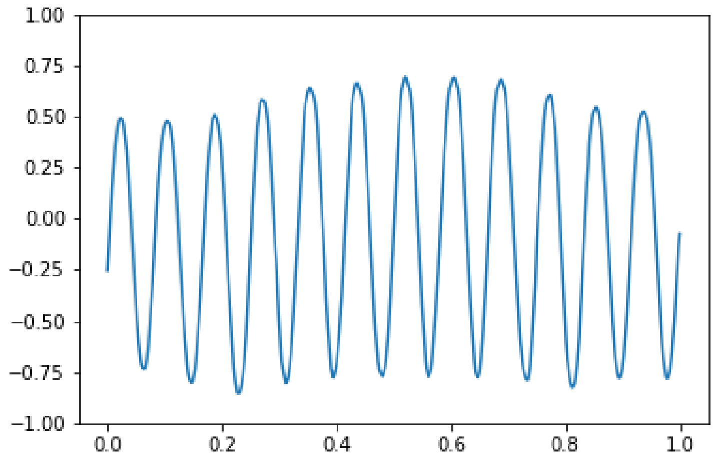
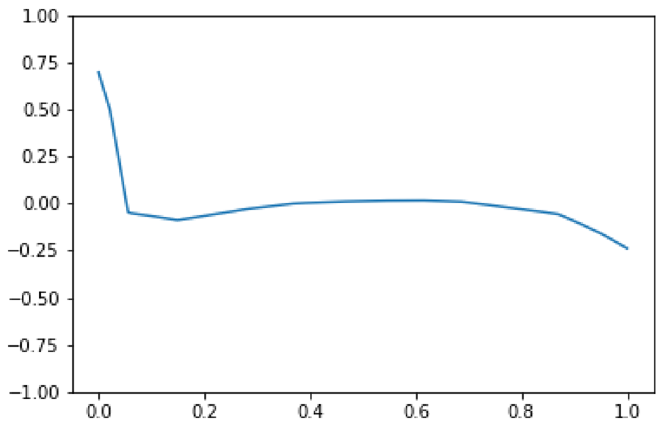
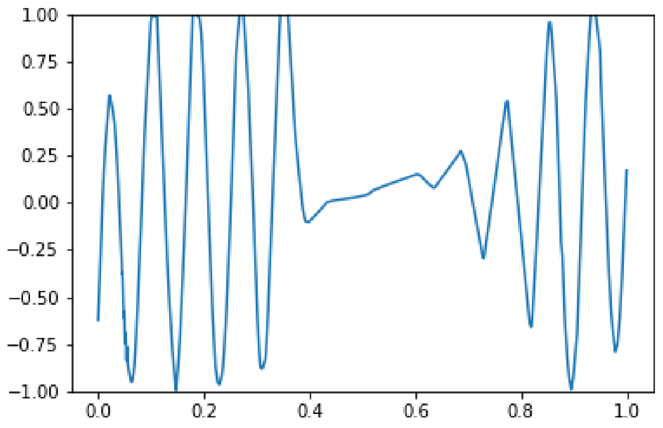


Fig. 5. Continued.

A



B



C

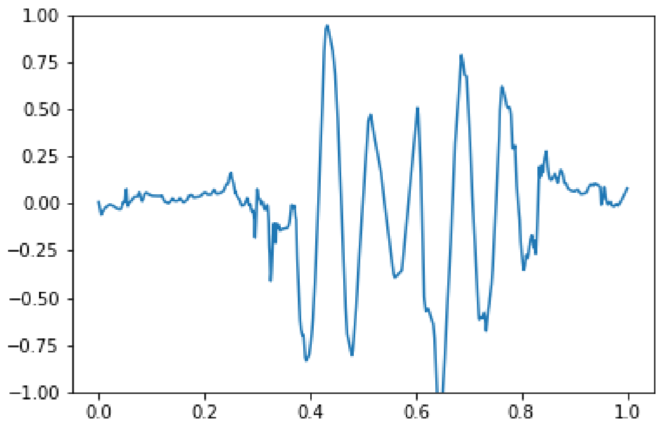
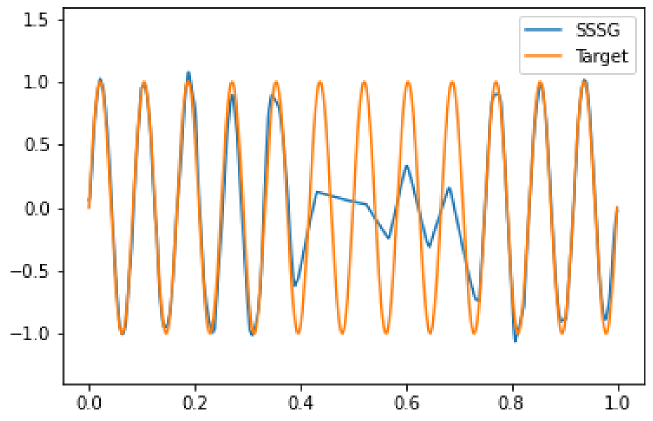
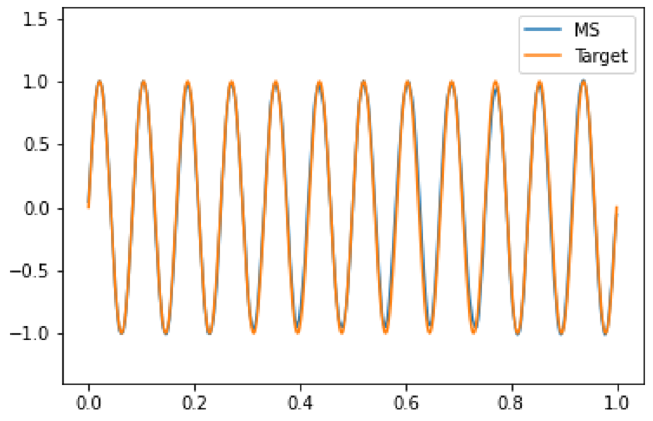


Fig. 6. For example 1 (noise free), a comparison of each grade in the multi-grade method, where each figure labeled A through C corresponds to grades one through three, respectively.

A



B



C

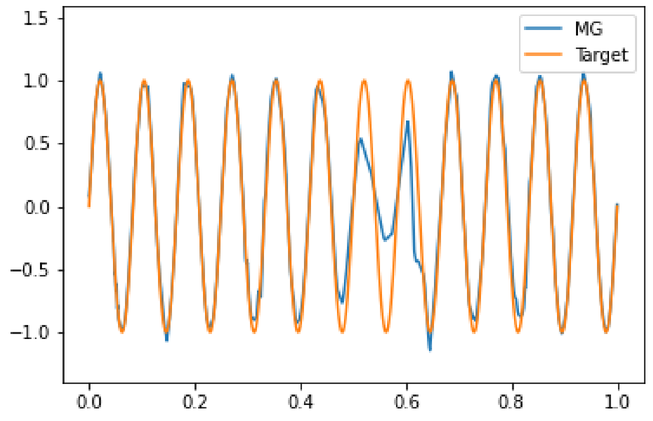


Fig. 7. For example 1 (noise free), the target function compared to the (A) single-scale, single grade model, (B) multi-scale model with eight scales, and (C) multi-grade model with three grades.

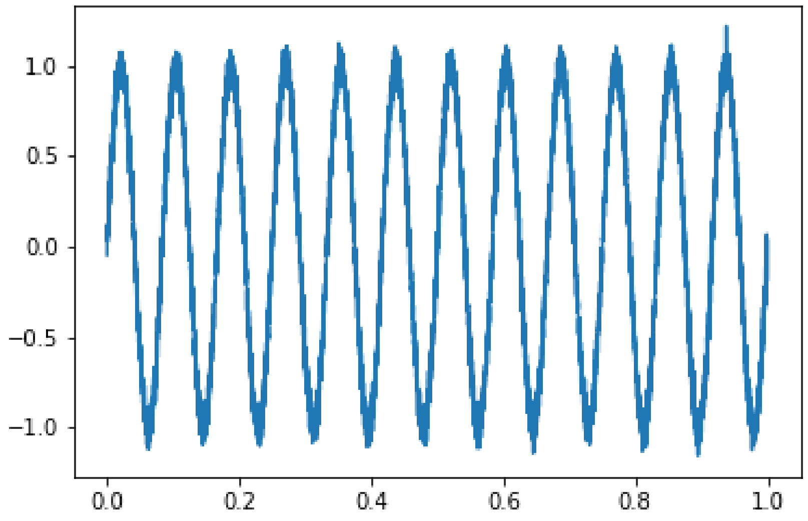


Fig. 8. The example 1 (noisy) target data set.

The training data for the noisy case are plotted in Figure 8. Numerical results for training for this noisy case are listed in Tables 6, 4, and 5. A corresponding figure is plotted in Figure 9.

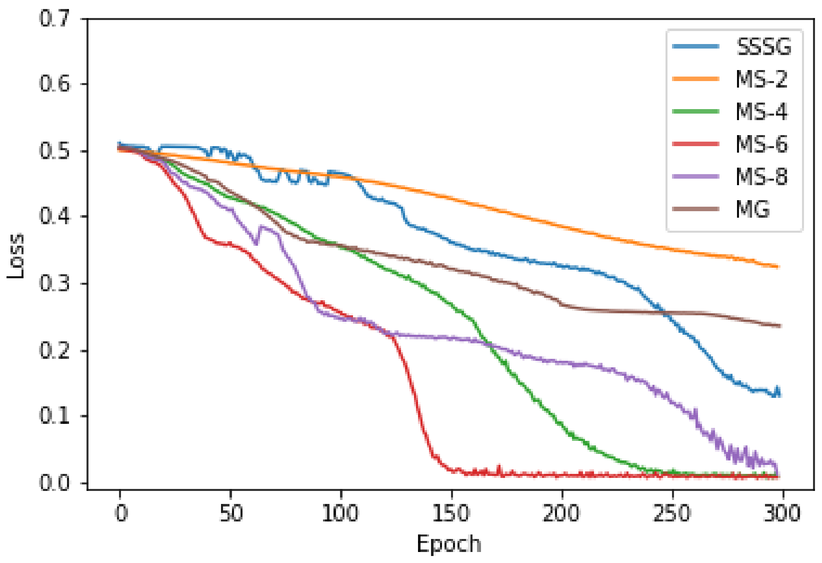


Fig. 9. For example 1 (noisy), the loss curves during training for the single-scale, single grade method, the multi-scale method with two, four, six, and eight scales, and the multi-grade method with 3 grades.

TABLE 4

For example 1 (noisy), the training time and accuracy multi-scale model with varying scales.

Scale	Training Time (seconds)	MSE
2	319.7761	0.3243
4	363.8955	0.0087
6	410.8616	0.0077
8	454.9396	0.0127

TABLE 5

For example 1 (noisy), the training time and accuracy for each grade of the multi-grade model.

Grade	Learning Rate	Batch Size	Epochs	Training Time (seconds)	MSE
1	10^{-4}	16	40	139.2121	0.4605
2	10^{-4}	16	160	152.4621	0.2724
3	10^{-4}	4	100	305.8536	0.2352

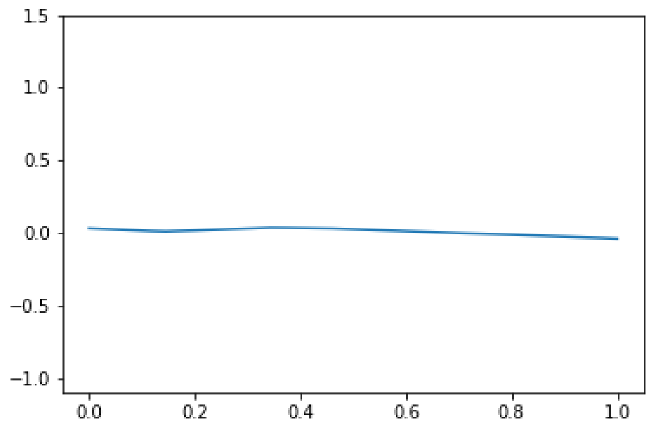
TABLE 6

For example 1 (noisy), a comparison of training time and accuracy for each model.

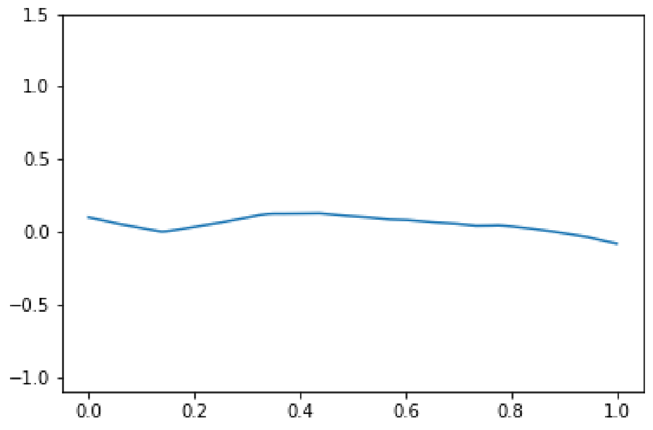
Method	Training Time (seconds)	MSE
Single-scale, single-grade	365.9769	0.1308
Multi-scale	454.9396	0.0127
Multi-grade	305.8536	0.2352

The prediction from each method for the noisy case is plotted in Figure 12. A plot of output from each scale of the multi-scale method with 8 scales is given in Figure 10. A plot of the output from each grade of the multi-grade method with 3 grades is given in Figure 11.

A



B



C

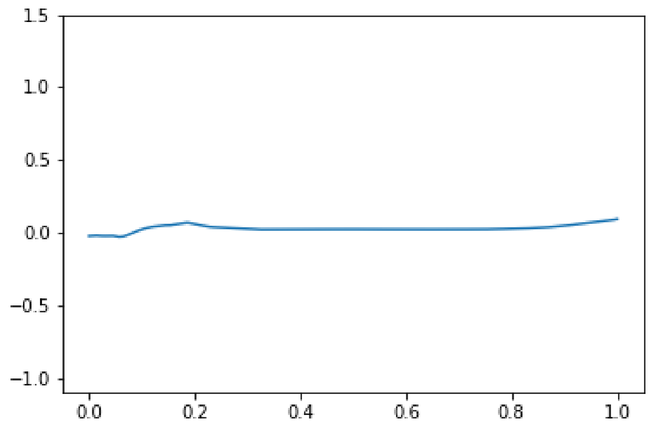
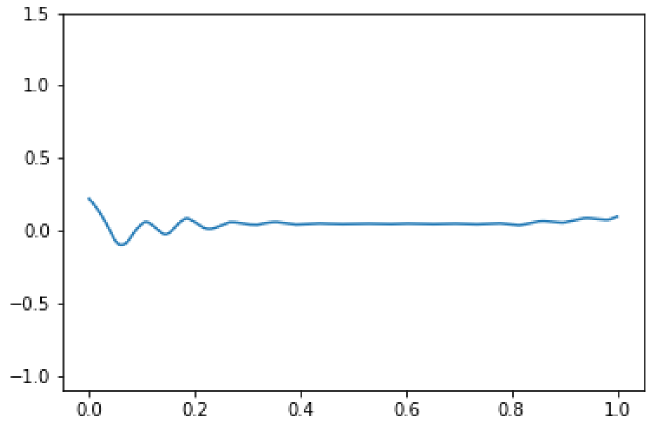
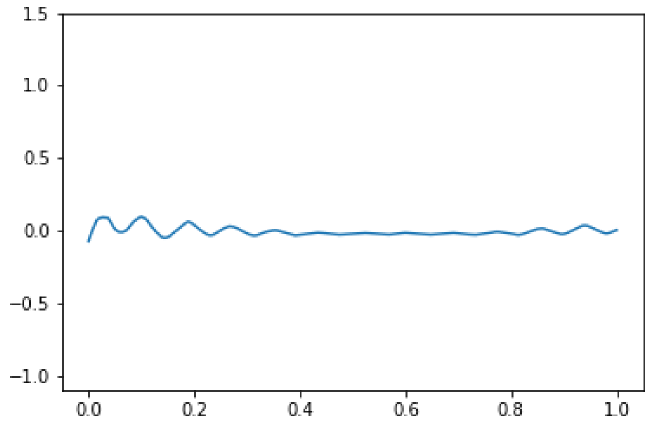


Fig. 10. For example 1 (noisy), the comparison of each scale in the multi-scale method, where each figure labeled A through H corresponds to scales one through eight, respectively.

D



E



F

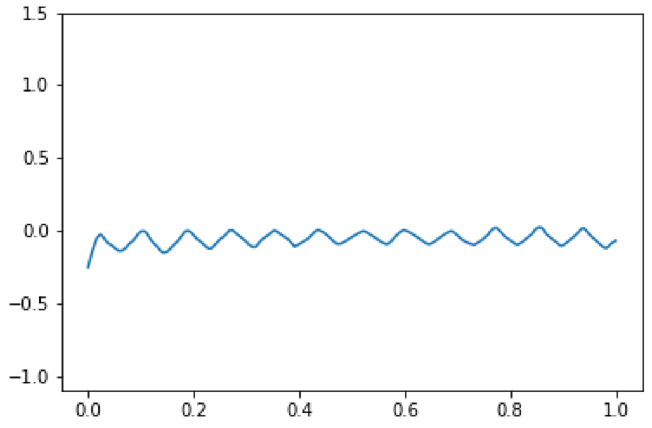
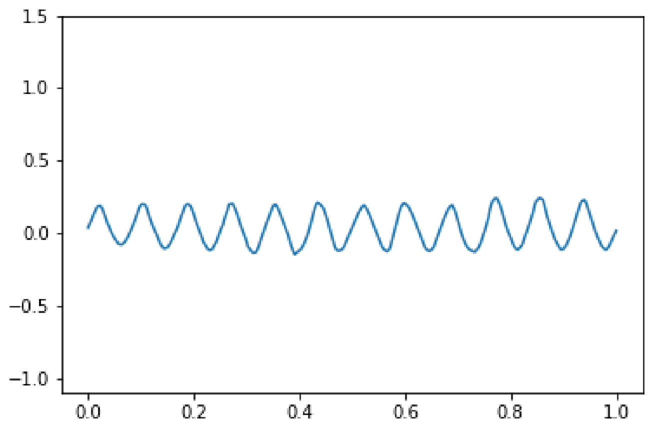


Fig. 10. Continued.

G



H

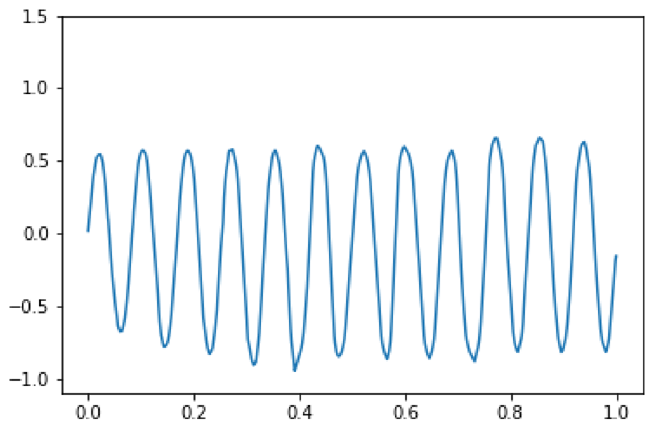
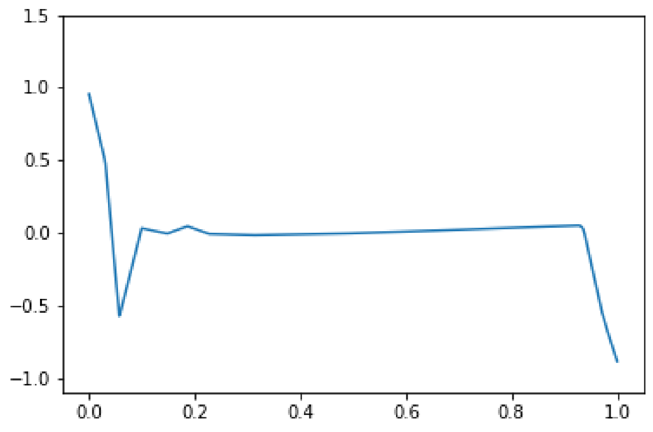
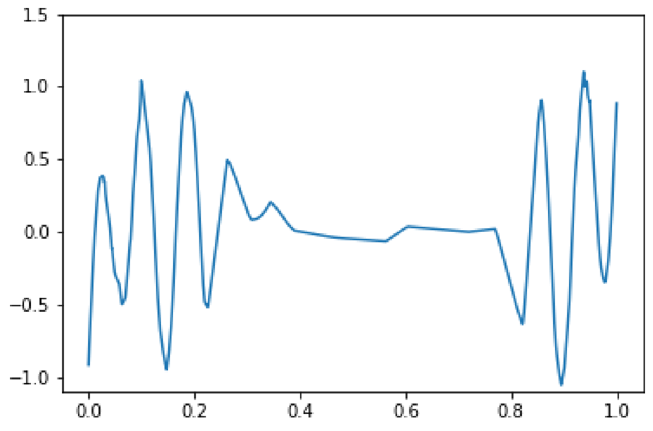


Fig. 10. Continued.

A



B



C

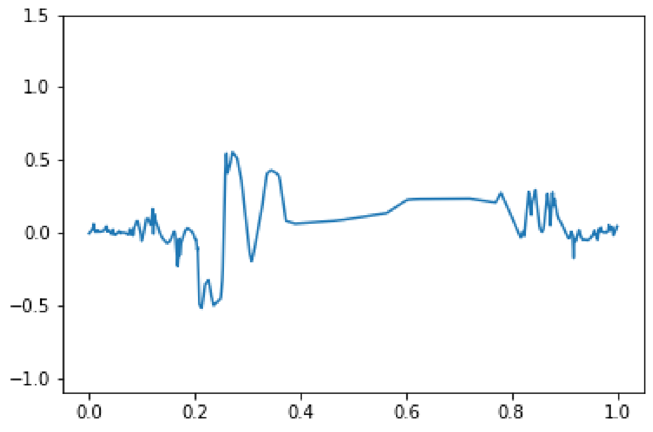
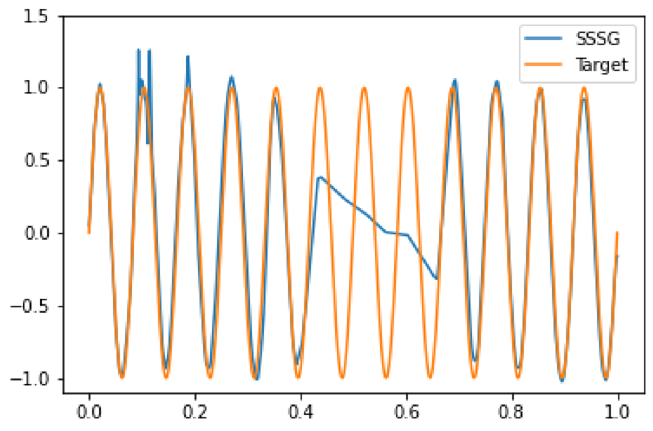
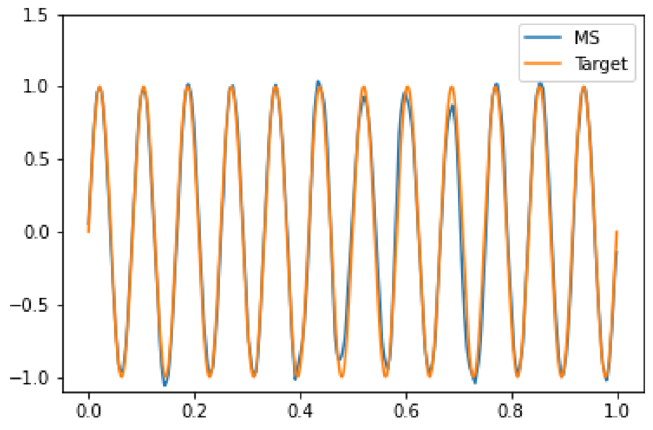


Fig. 11. For example 1 (noisy), the comparison of each grade in the multi-grade method, where each figure labeled A through C corresponds to grades one through three, respectively.

A



B



C

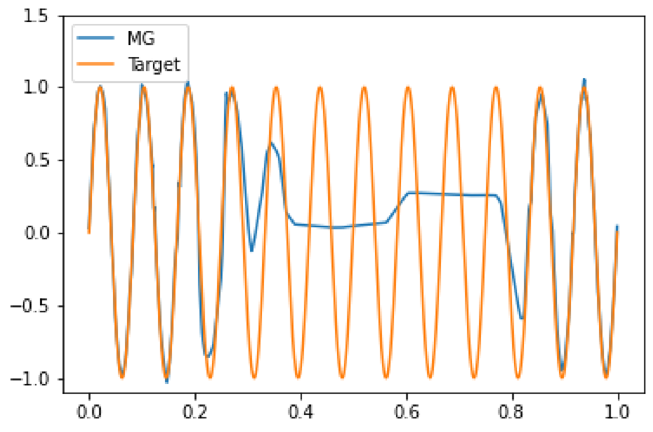


Fig. 12. For example 1 (noisy), the target function compared to the (A) single-scale, single grade model, (B) multi-scale model with eight scales, and (C) multi-grade model with three grades.

3.2 EXAMPLE 2

Consider the task of approximating the oscillatory function

$$f(x) = x \sin(24\pi x), \quad x \in [0,1] \tag{3.3}$$

For training all models, 300 epochs are used. The single-scale, single-grade network and the multi-scale networks use a batch size of 16 and an initial learning rate of $10^{-}4$. The number of epochs, batch size, and initial learning rate per grade for the multi-grade model are described in Table 8 for the noise-free case and Table 11 for the noisy case.

The training data for the noise-free case are plotted in Figure 13.

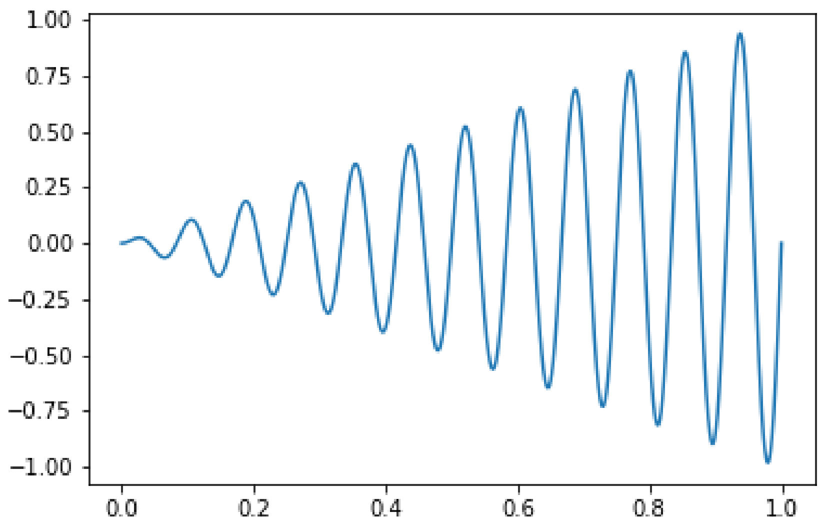


Fig. 13. The example 2 (noise free) target data set.

Numerical results for training for this noise-free case are listed in Tables 9, 7, and 8. A corresponding figure is plotted in Figure 14.

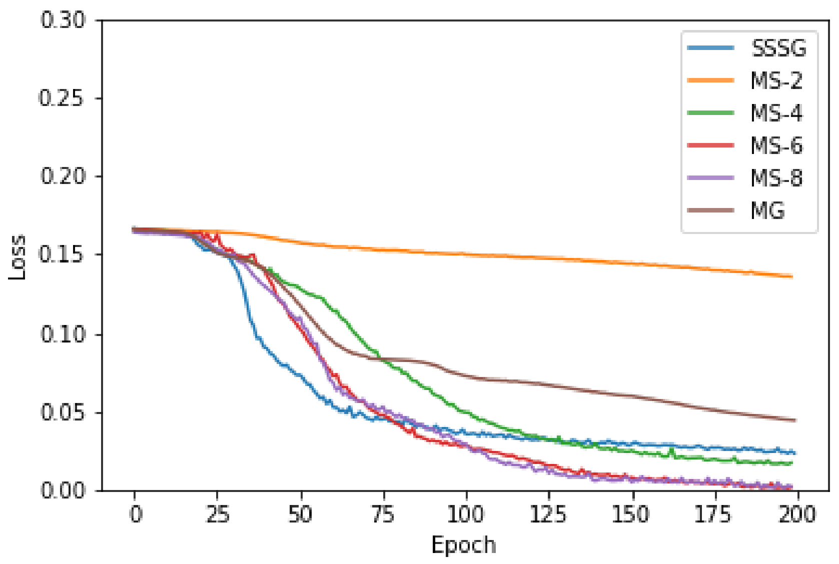


Fig. 14. For example 2 (noise free), the loss curves during training for the single-scale, single grade method, the multi-scale method with two, four, six, and eight scales, and the multi-grade method with 3 grades.

TABLE 7

For example 2 (noise free), the training time and accuracy multi-scale model with varying scales.

Scale	Training Time (seconds)	MSE
2	297.3770	0.0947
4	402.8942	0.0065
6	409.0968	0.0006
8	438.5784	0.0005

TABLE 8

For example 2 (noise free), thee training time and accuracy for each grade of the multi-grade model.

Grade	Learning Rate	Batch Size	Epochs	Training Time (seconds)	MSE
1	10^{-4}	16	40	39.7909	0.4829
2	10^{-4}	16	160	150.3363	0.1863
3	10^{-4}	4	100	311.1016	0.0429

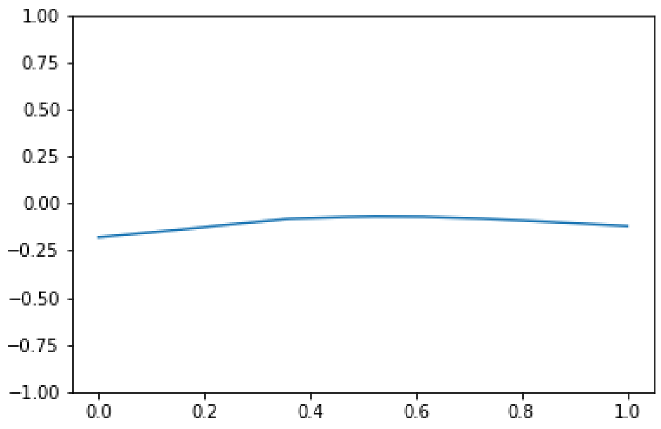
TABLE 9

For example 2 (noise free), a comparison of training time and accuracy for each model.

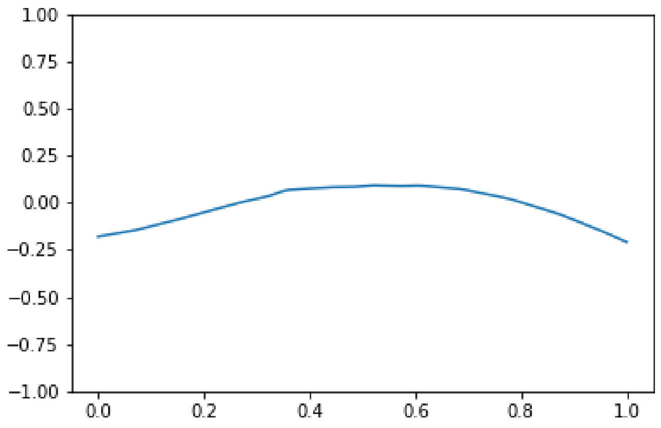
Method	Training Time (seconds)	MSE
Single-scale, single-grade	367.9119	0.0020
Multi-scale	438.5784	0.0005
Multi-grade	501.2288	0.0429

The prediction from each method for the noise-free case is plotted in Figure 17. A plot of output from each scale of the multi-scale method with 8 scales is given in Figure 15. A plot of the output from each grade of the multi-grade method with 3 grades is given in Figure 16.

A



B



C

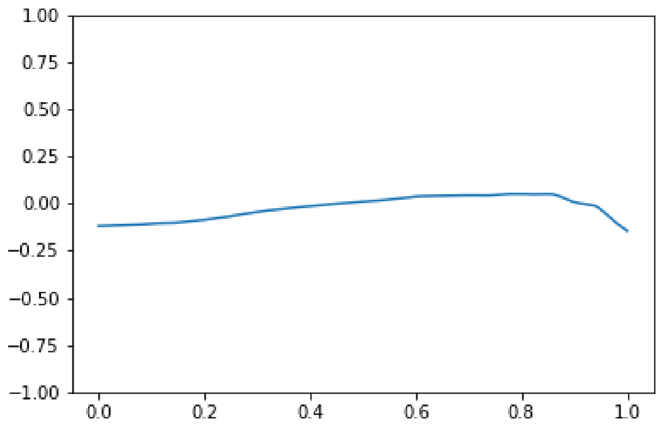
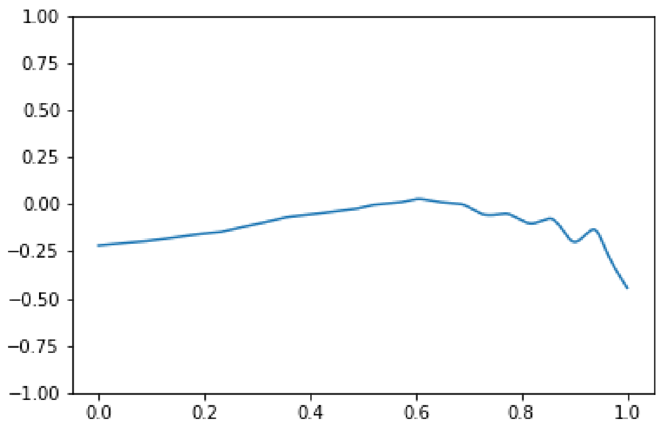
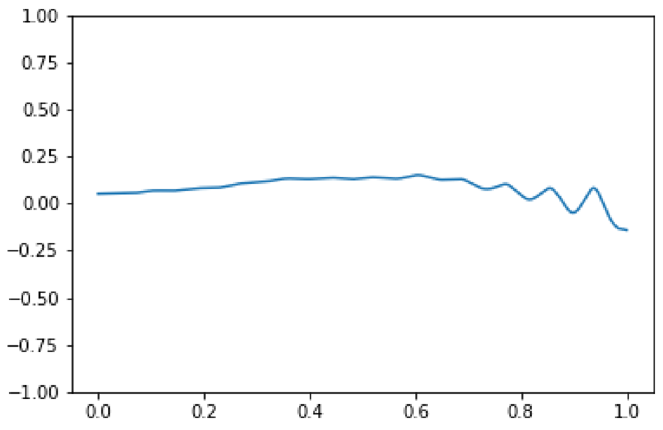


Fig. 15. For example 2 (noise free), the comparison of each scale in the multi-scale method, where each figure labeled A through H corresponds to scales one through eight, respectively.

D



E



F

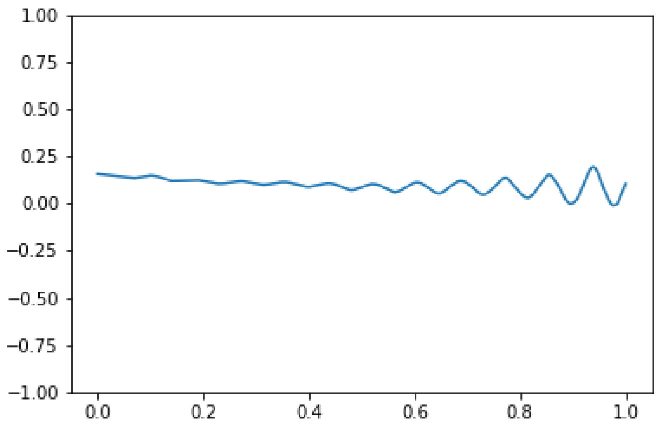
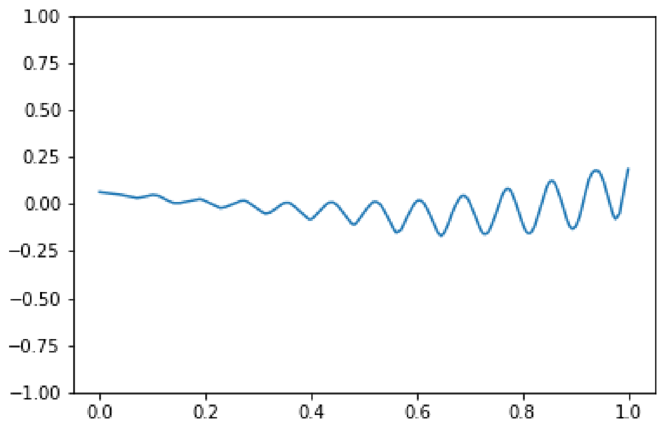


Fig. 15. Continued.

G



H

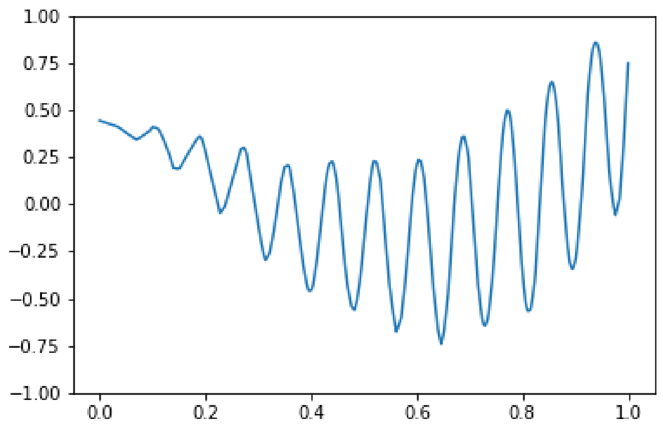


Fig. 15. Continued.

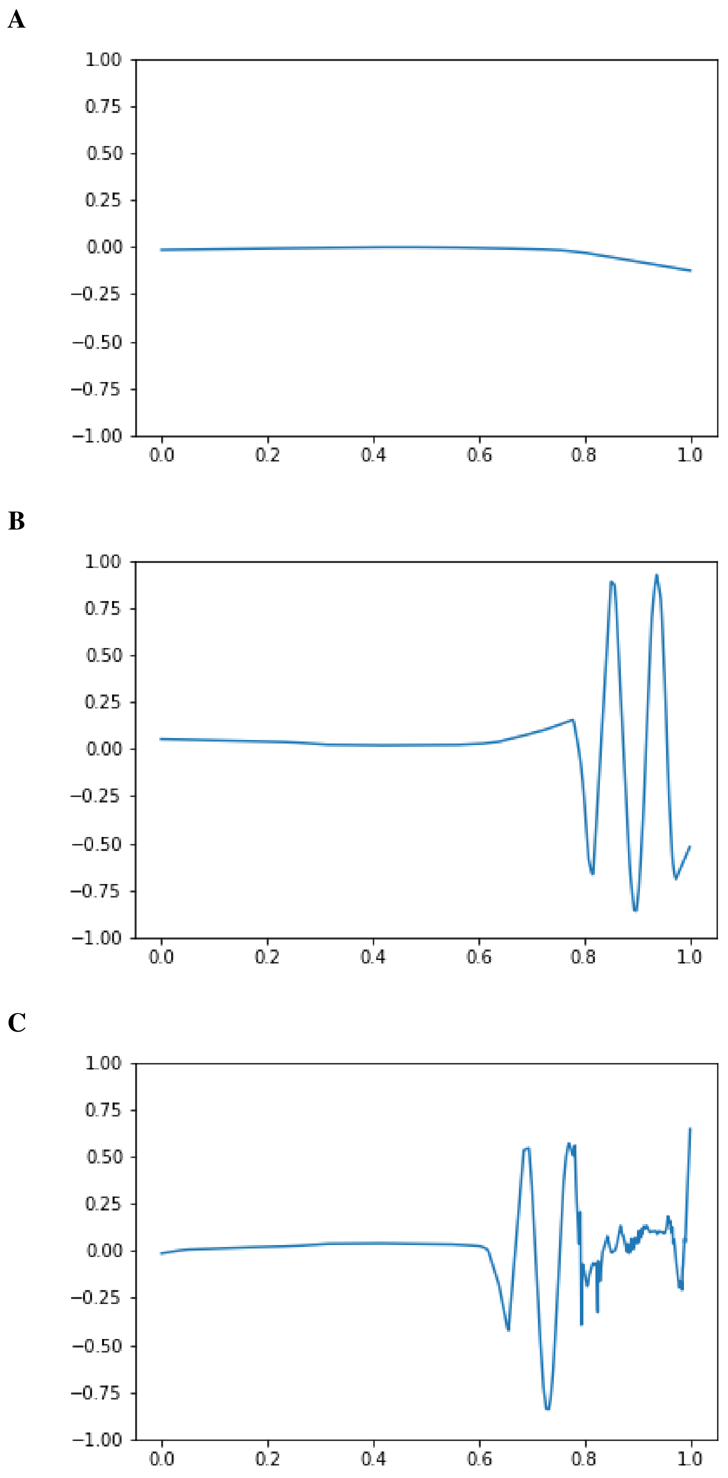
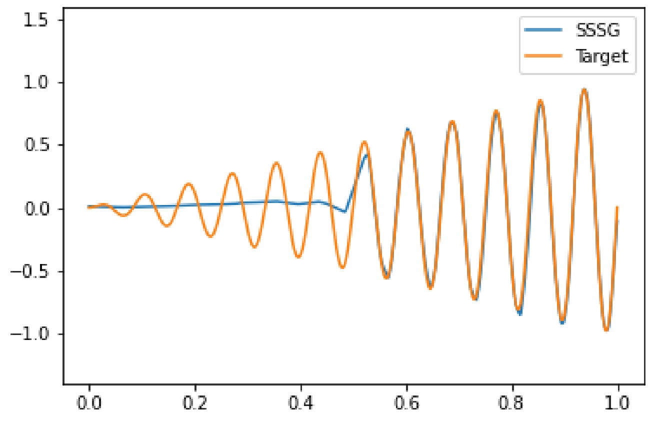
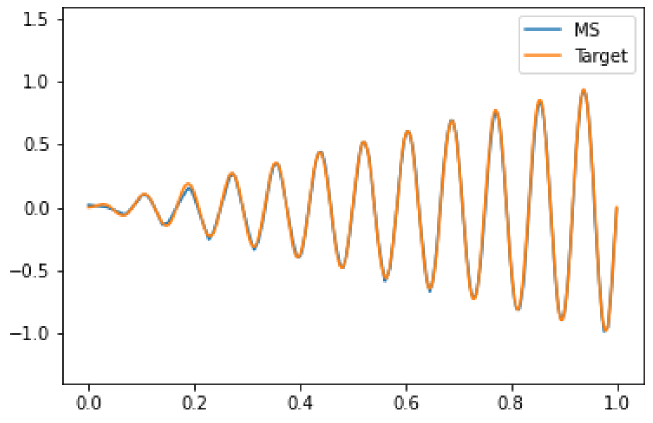


Fig. 16. For example 2 (noise free), a comparison of each grade in the multi-grade method, where each figure labeled A through C corresponds to grades one through three, respectively.

A



B



C

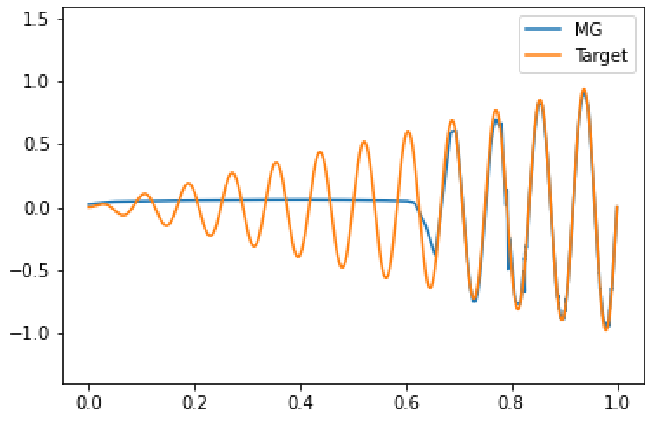


Fig. 17. For example 2 (noise free), the target function compared to the (A) single-scale, single grade model, (B) multi-scale model with eight scales, and (C) multi-grade model with three grades.

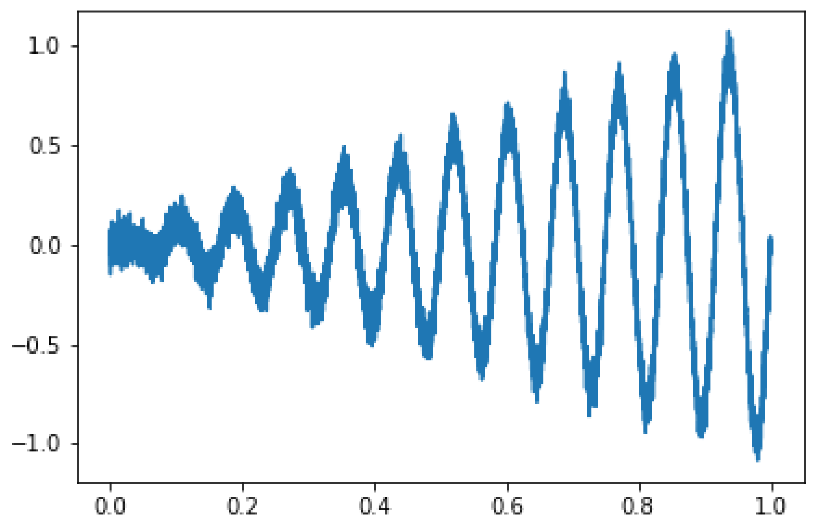


Fig. 18. The example 2 (noisy) target data set.

The training data for the noisy case are plotted in Figure 18. Numerical results for training for this noisy case are listed in Tables 12, 10, and 11. A corresponding figure is plotted in Figure 19.

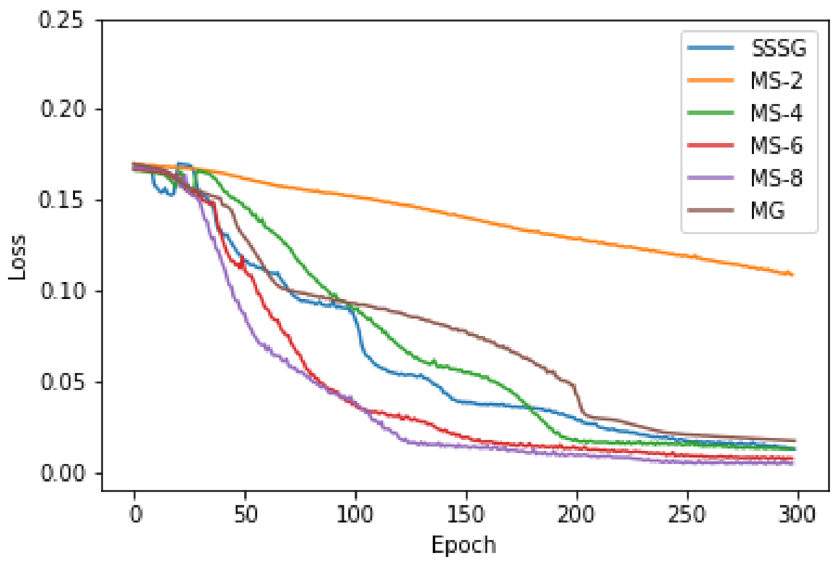


Fig. 19. For example 2 (noisy), the loss curves during training for the single-scale, single grade method, the multi-scale method with two, four, six, and eight scales, and the multi-grade method with 3 grades.

TABLE 10

For example 2 (noisy), the training time and accuracy multi-scale model with varying scales.

Scale	Training Time (seconds)	MSE
2	290.0700	0.1087
4	396.3188	0.0127
6	409.2024	0.0073
8	429.1318	0.0045

TABLE 11

For example 2 (noisy), the training time and accuracy for each grade of the multi-grade model.

Grade	Learning Rate	Batch Size	Epochs	Training Time (seconds)	MSE
1	10^{-4}	4	40	145.8970	0.1509
2	10^{-4}	16	160	156.8009	0.0479
3	10^{-4}	4	100	325.1570	0.0171

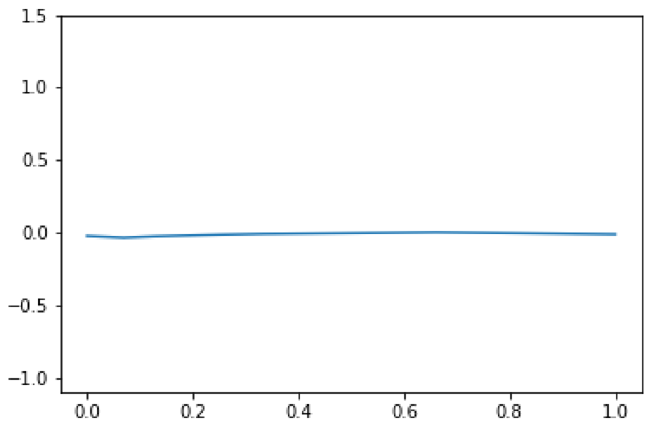
TABLE 12

For example 2 (noisy), the comparison of training time and accuracy for each model.

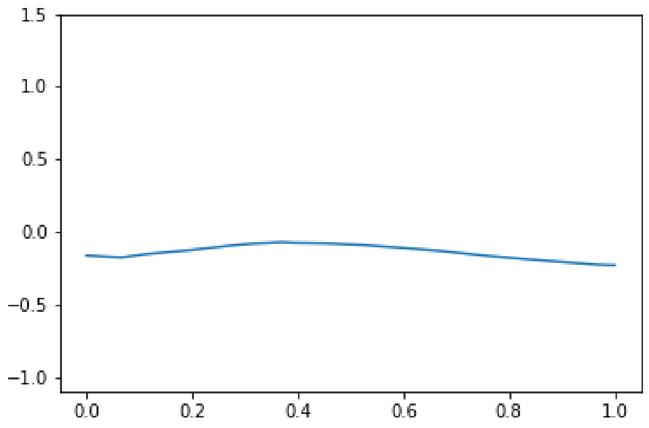
Method	Training Time (seconds)	MSE
Single-scale, single-grade	403.5050	0.0125
Multi-scale	429.1318	0.0045
Multi-grade	627.8549	0.0171

The prediction from each method for the noisy case is plotted in Figure 22. A plot of output from each scale of the multi-scale method with 8 scales is given in Figure 20. A plot of the output from each grade of the multi-grade method with 3 grades is given in Figure 21.

A



B



C

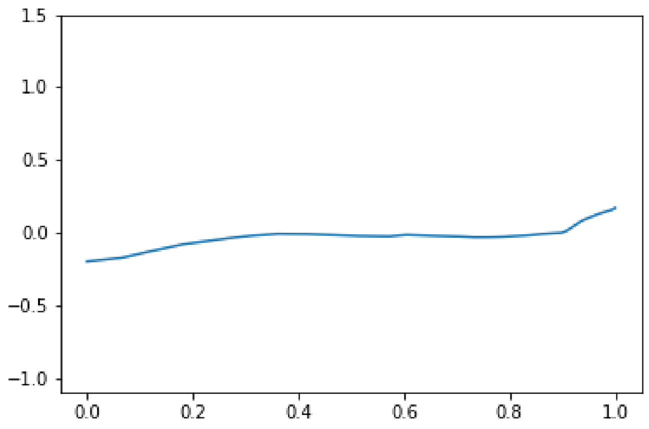
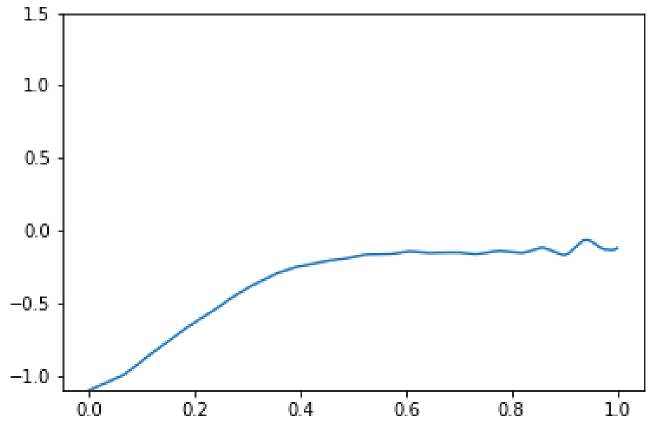
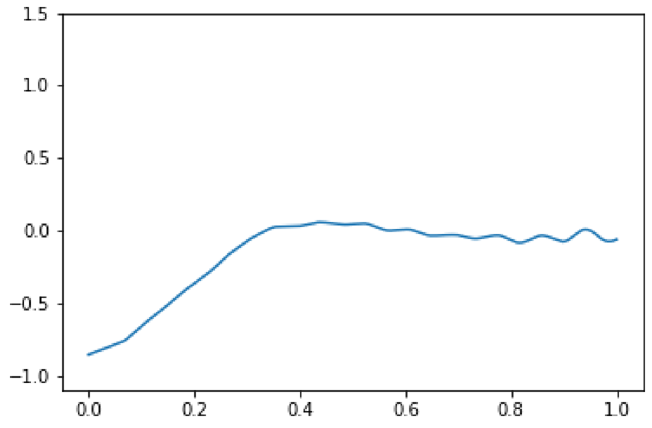


Fig. 20. For example 2 (noisy), a comparison of each scale in the multi-scale method, where each figure labeled A through H corresponds to scales one through eight, respectively.

D



E



F

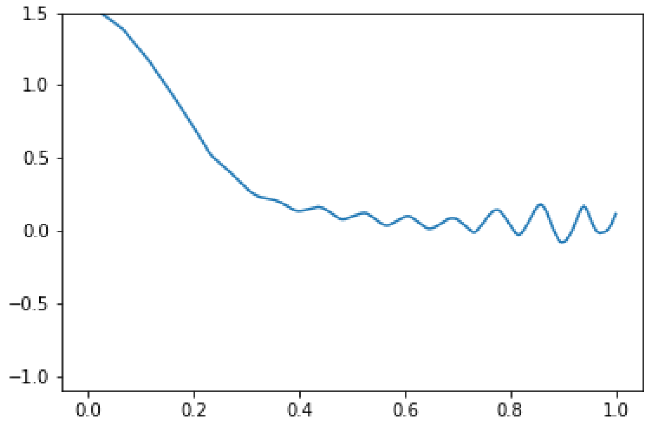
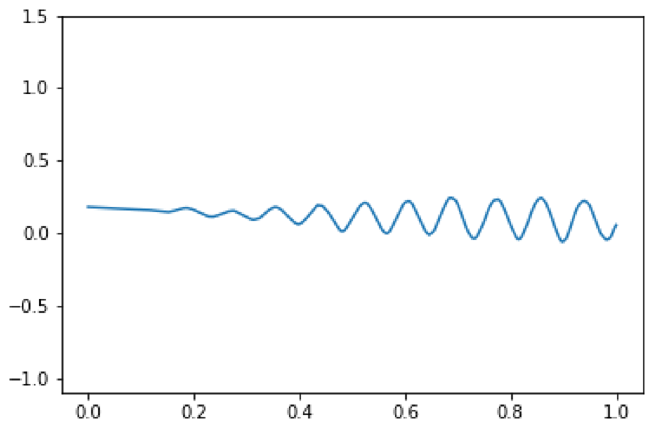


Fig. 20. Continued.

G



H

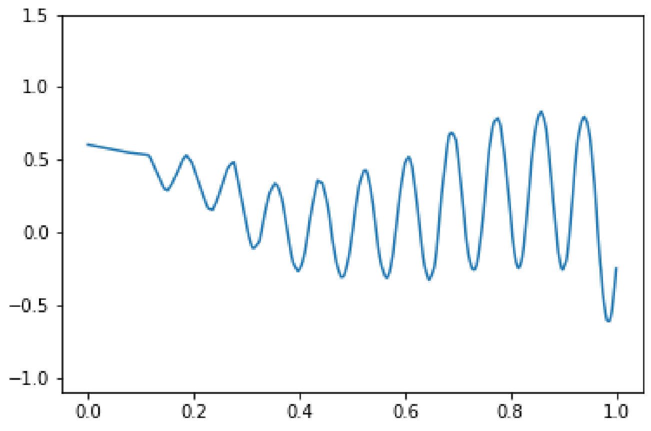
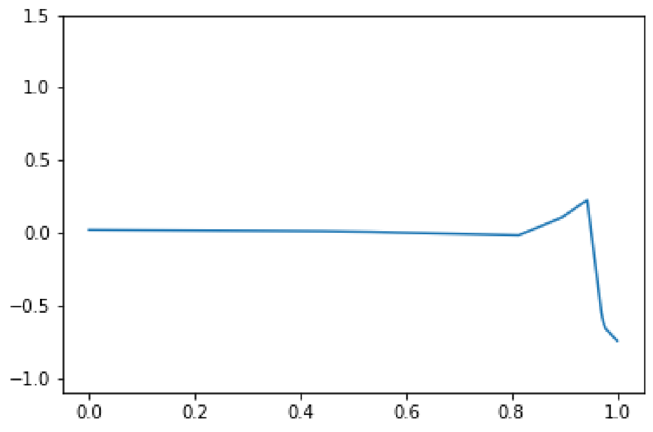
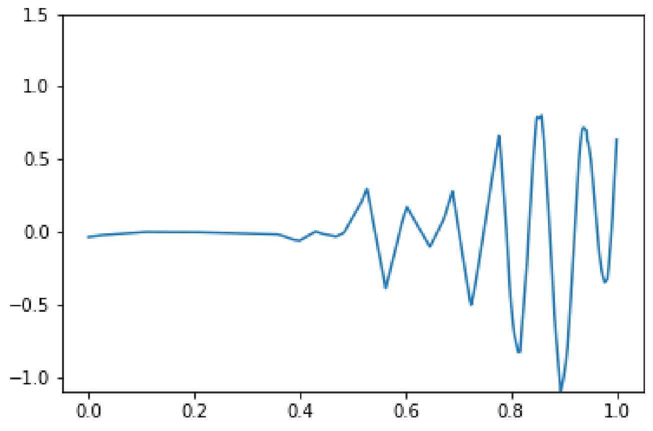


Fig. 20. Continued.

A



B



C

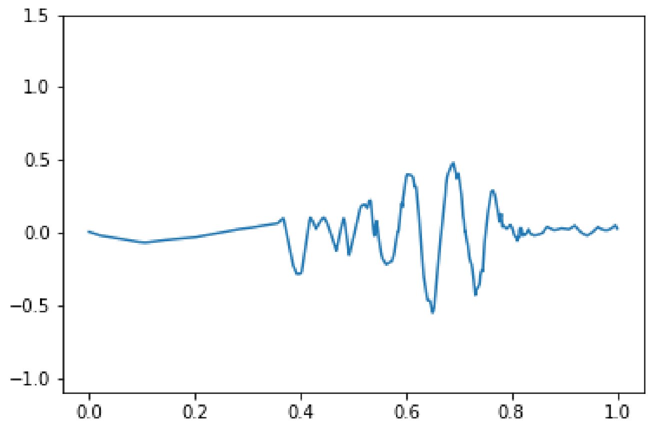


Fig. 21. For example 2 (noisy), a comparison of each grade in the multi-grade method, where each figure labeled A through C corresponds to grades one through three, respectively.

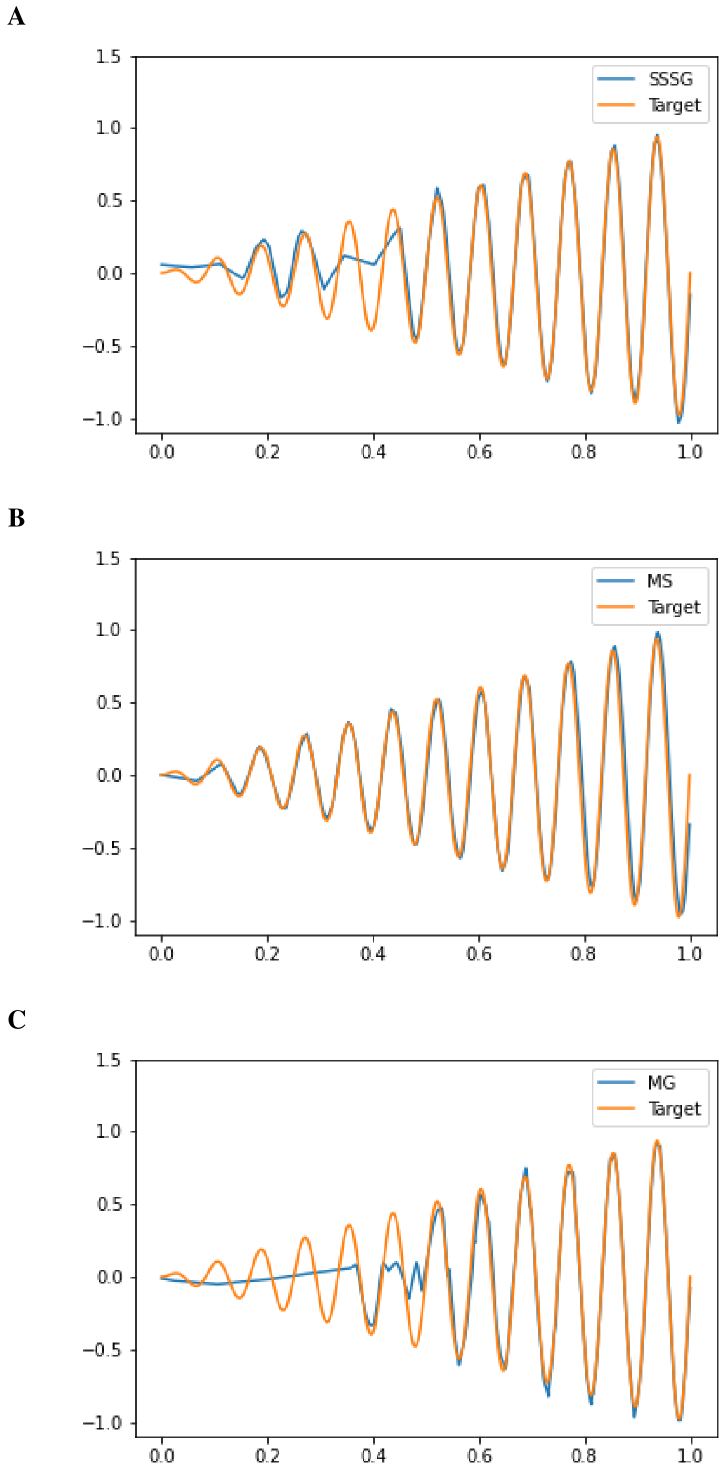


Fig. 22. For example 2 (noisy), the target function compared to the (A) single-scale, single grade model, (B) multi-scale model with eight scales, and (C) multi-grade model with three grades.

3.3 EXAMPLE 3

Consider the task of approximating the nonsmooth function

$$f(x) = (x+1)(f_4 \circ f_3 \circ f_2 \circ f_1)(x), \quad x \in [-1,1], \tag{3.4}$$

where

$$\begin{aligned} f_1(x) &= |\cos(\pi(x-0.3)) - 0.7|, \\ f_2(x) &= |\cos(2\pi(x-0.5)) - 0.5|, \\ f_3(x) &= -|x-1.3| + 1.3, \\ f_4(x) &= -|x-0.9| + 0.9. \end{aligned}$$

For training all models, 250 epochs are used. The single-scale, single-grade network and the multi-scale networks use a batch size of 16 and an initial learning rate of $10^{-}4$. The number of epochs, batch size, and initial learning rate per grade for the multi-grade model are described in Table 14 for the noise-free case and Table 17 for the noisy case.

The training data for the noise-free case are plotted in Figure 23.

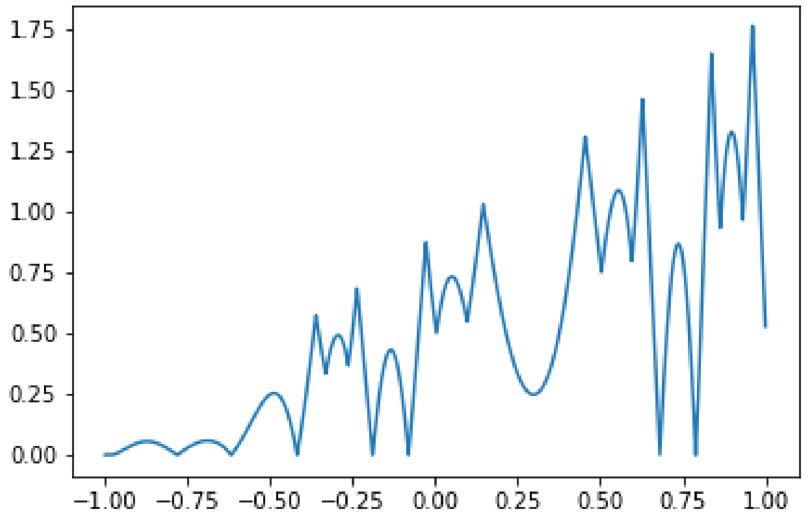


Fig. 23. The example 3 (noise free) target data set.

Numerical results for training for this noise-free case are listed in Tables 15, 13, and 14. A corresponding figure is plotted in Figure 24.

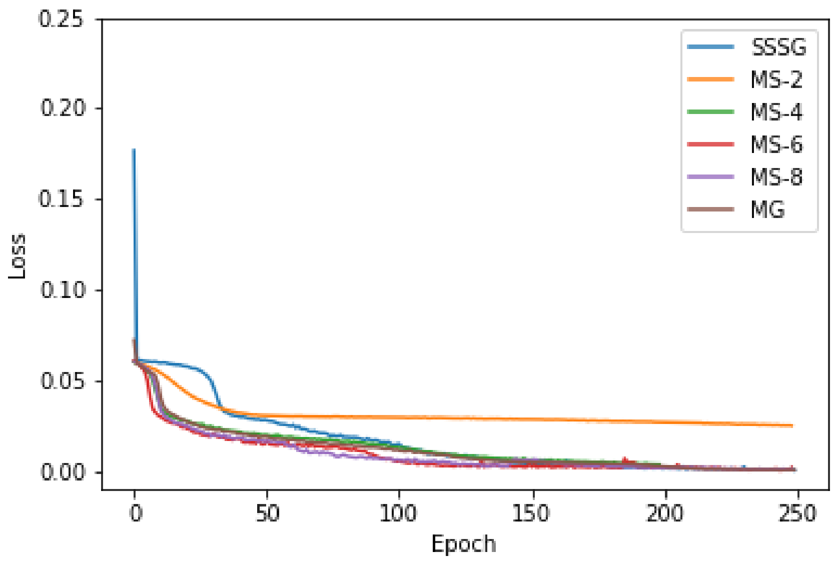


Fig. 24. For example 3 (noise free), the loss curves during training for the single-scale, single grade method, the multi-scale method with two, four, six, and eight scales, and the multi-grade method with 3 grades.

The prediction from each method for the noise-free case is plotted in Figure 27. A plot of output from each scale of the multi-scale method with 8 scales is given in Figure 25. A plot of the output from each grade of the multi-grade method with 3 grades is given in Figure 26.

The training data for the noisy case are plotted in Figure 28. Numerical results for training for this noisy case are listed in Tables 18, 16, and 17. A corresponding figure is plotted in Figure 29.

TABLE 13

For example 3 (noise free), the training time and accuracy multi-scale model with varying scales.

Scale	Training Time (seconds)	MSE
2	246.9515	0.0249
4	150.3363	0.1863
6	256.1678	0.0005
8	322.9416	0.0003

TABLE 14

For example 3 (noise free), the training time and accuracy for each grade of the multi-grade model.

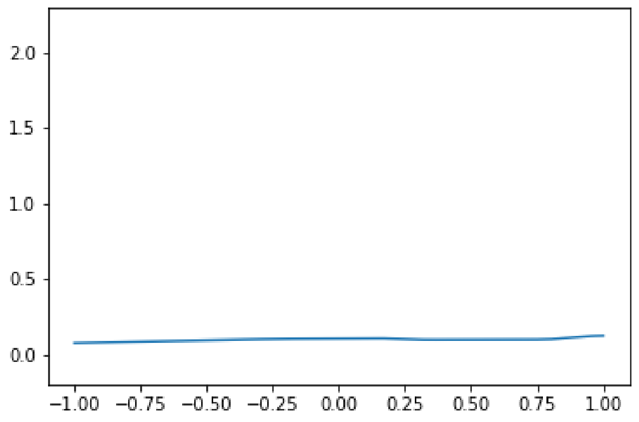
Grade	Learning Rate	Batch Size	Epochs	Training Time (seconds)	MSE
1	10^{-4}	4	5	19.8238	0.0563
2	10^{-4}	16	115	115.9054	0.0083
3	10^{-4}	4	130	424.5047	0.0005

TABLE 15

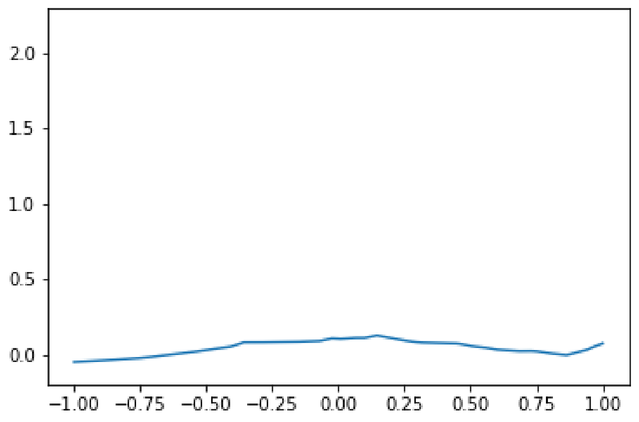
For example 3 (noise free), a comparison of training time and accuracy for each model.

Method	Training Time (seconds)	MSE
Single-scale, single-grade	306.6797	0.0003
Multi-scale	322.9416	0.0003
Multi-grade	560.2339	0.0429

A



B



C

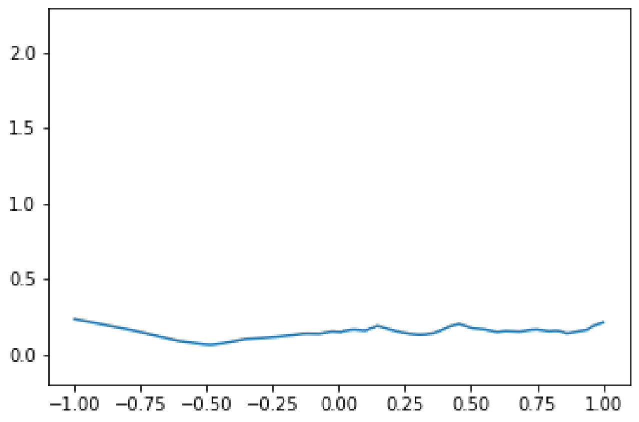
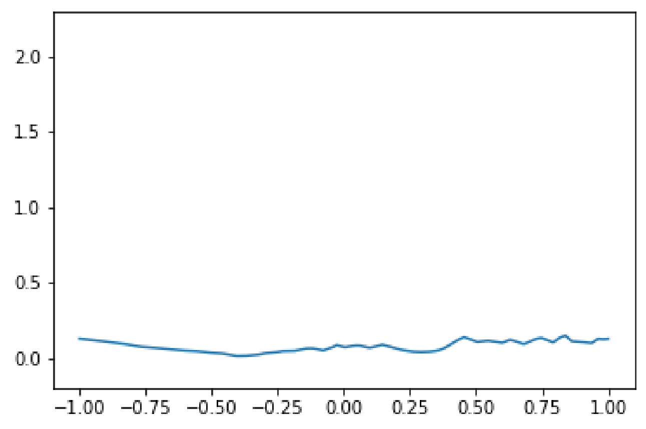
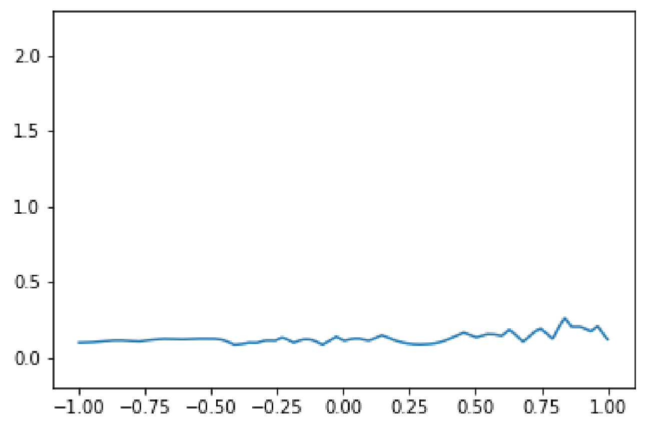


Fig. 25. For example 3 (noise free), a comparison of each scale in the multi-scale method, where each figure labeled A through H corresponds to scales one through eight, respectively.

D



E



F

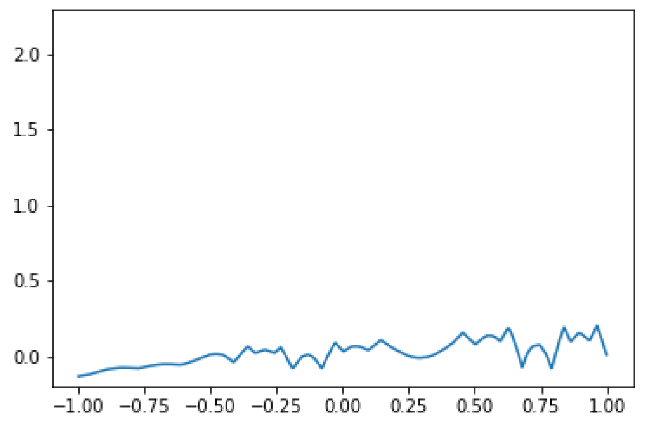
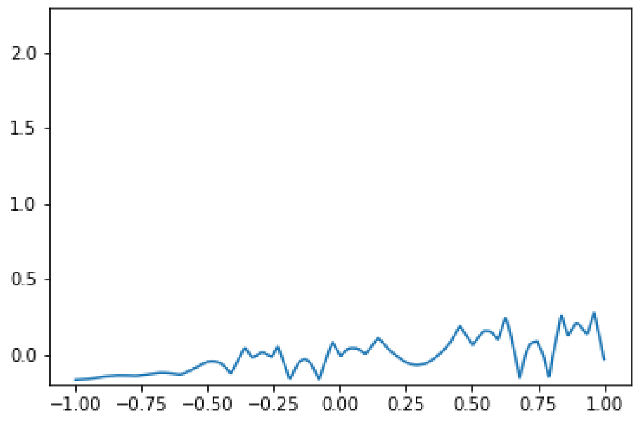


Fig. 25. Continued.

G



H

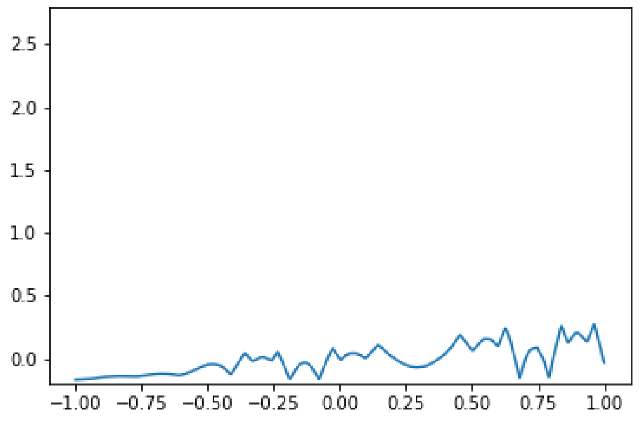


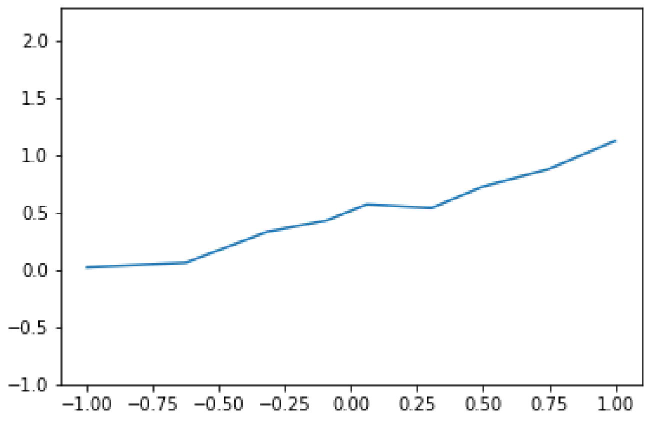
Fig. 25. Continued.

TABLE 16

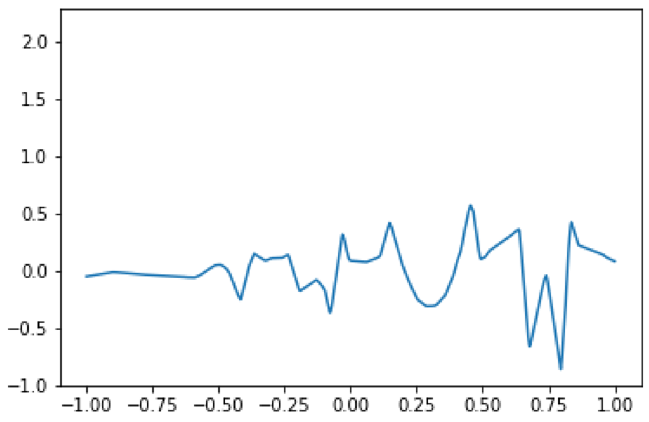
For example 3 (noisy), the training time and accuracy multi-scale model with varying scales.

Scale	Training Time (seconds)	MSE
2	259.7693	0.0272
4	313.5776	0.0056
6	371.6165	0.0036
8	366.1987	0.0042

A



B



C

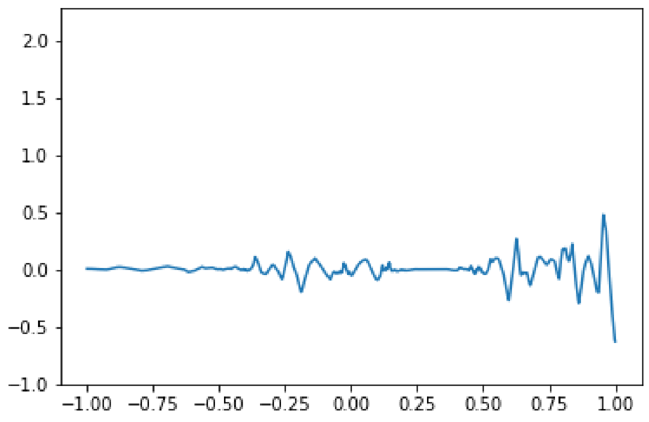
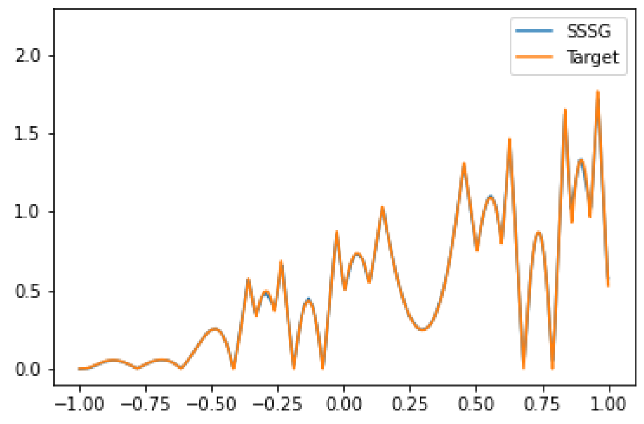
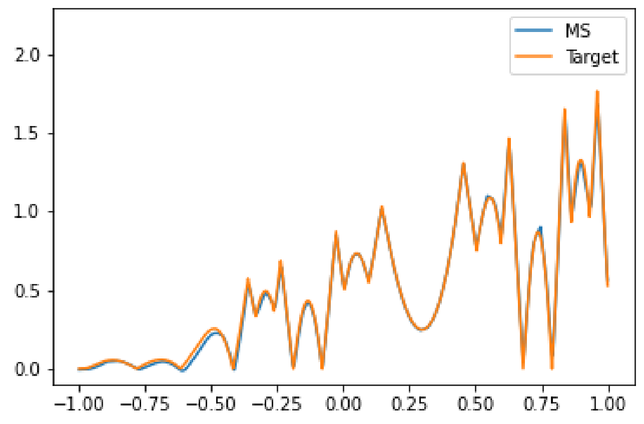


Fig. 26. For example 3 (noise free), a comparison of each grade in the multi-grade method, where each figure labeled A through C corresponds to grades one through three, respectively.

A



B



C

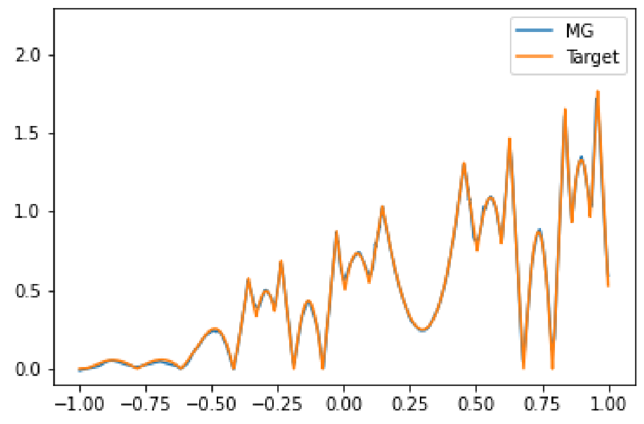


Fig. 27. For example 3 (noise free), the target function compared to the (A) single-scale, single grade model, (B) multi-scale model with eight scales, and (C) multi-grade model with three grades.

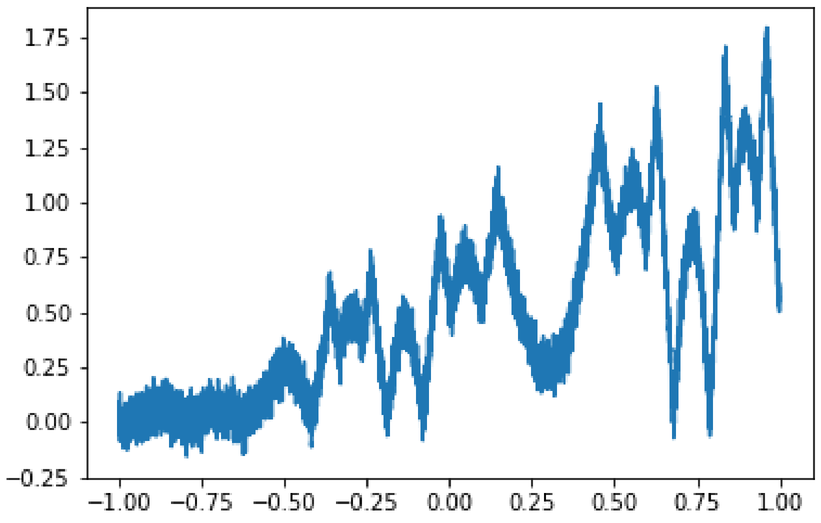


Fig. 28. The example 3 (noisy) target data set.

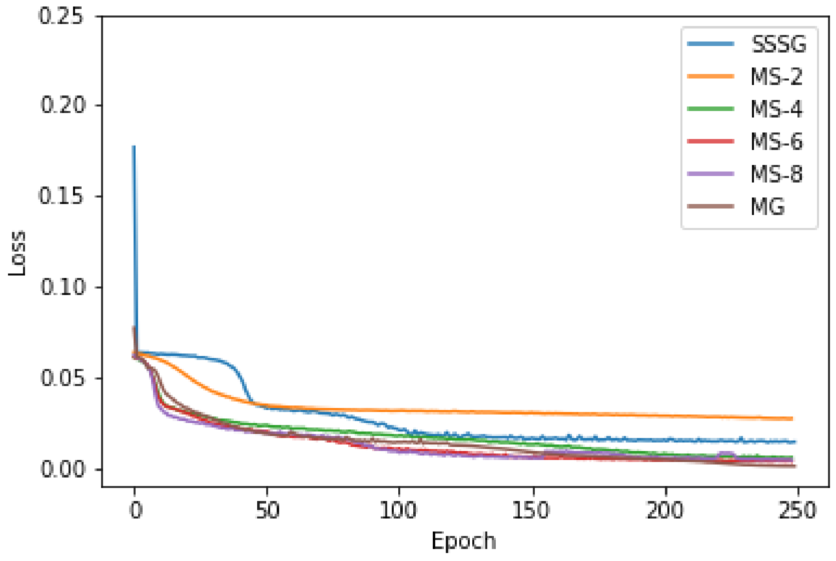


Fig. 29. For example 3 (noisy), the loss curves during training for the single-scale, single grade method, the multi-scale method with two, four, six, and eight scales, and the multi-grade method with 3 grades.

TABLE 17

For example 3 (noisy), the training time and accuracy for each grade of the multi-grade model.

Grade	Learning Rate	Batch Size	Epochs	Training Time (seconds)	MSE
1	10^{-4}	4	40	19.7853	0.0585
2	10^{-4}	16	115	116.1317	0.0127
3	10^{-4}	4	130	431.3353	0.0009

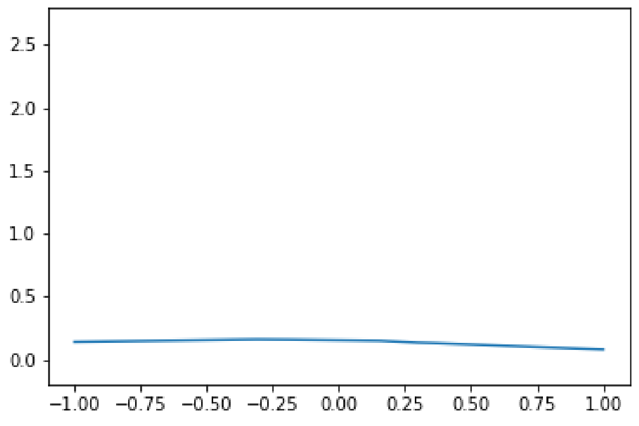
TABLE 18

For example 3 (noisy), a comparison of training time and accuracy for each model.

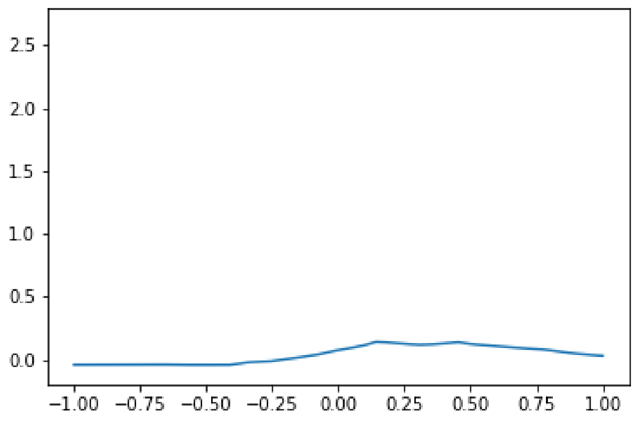
Method	Training Time (seconds)	MSE
Single-scale, single-grade	322.5063	0.0143
Multi-scale	366.1987	0.0042
Multi-grade	567.2523	0.0009

The prediction from each method for the noisy case is plotted in Figure 32. A plot of output from each scale of the multi-scale method with 8 scales is given in Figure 30. A plot of the output from each grade of the multi-grade method with 3 grades is given in Figure 31.

A



B



C

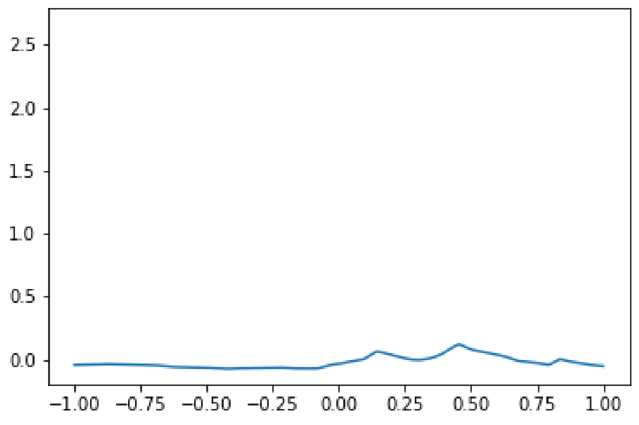
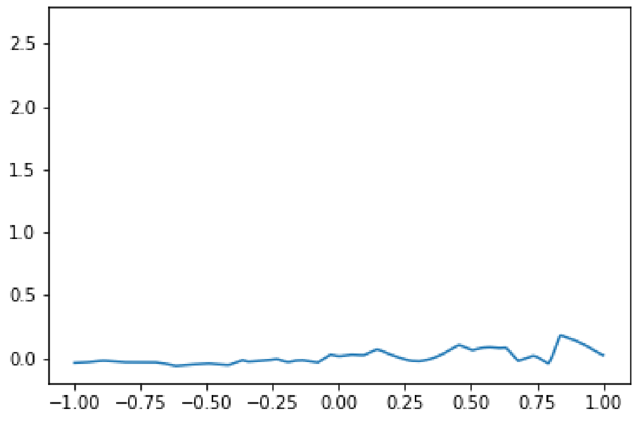
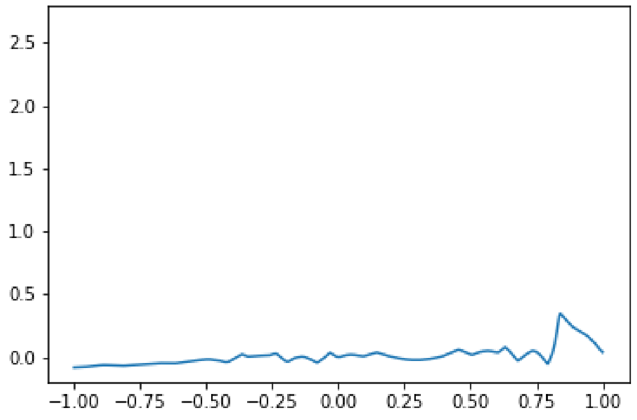


Fig. 30. For example 3 (noisy), a comparison of each scale in the multi-scale method, where each figure labeled A through H corresponds to scales one through eight, respectively.

D



E



F

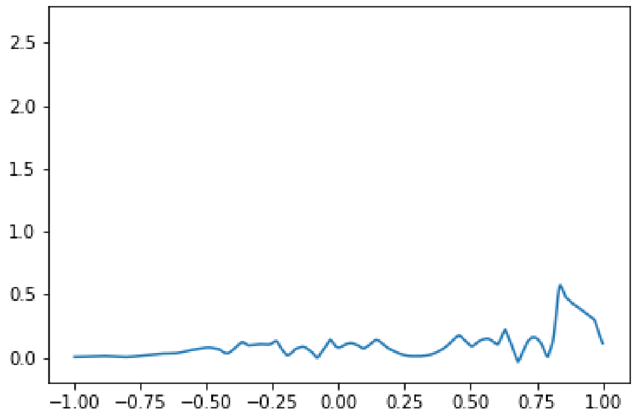
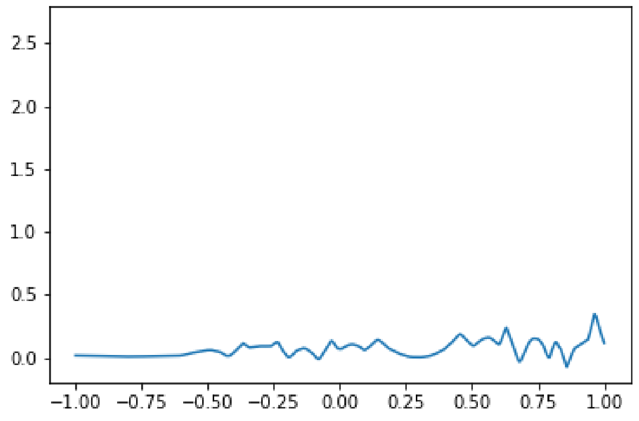


Fig. 30. Continued.

G



H

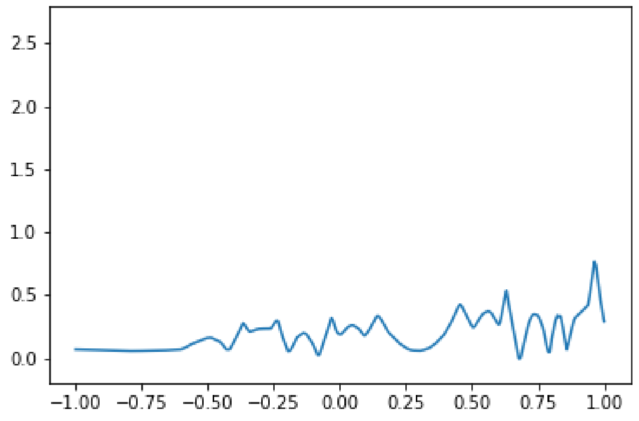
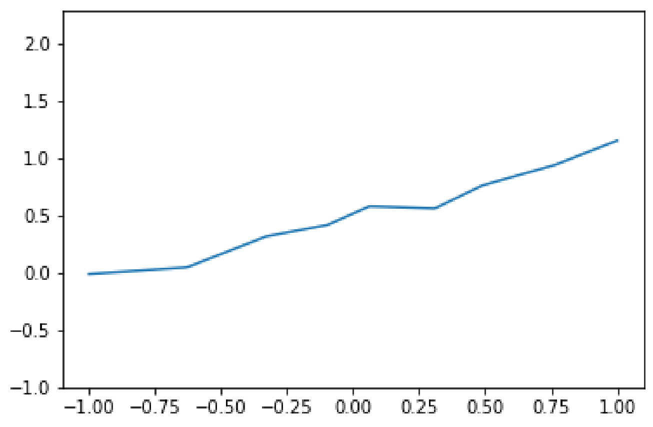
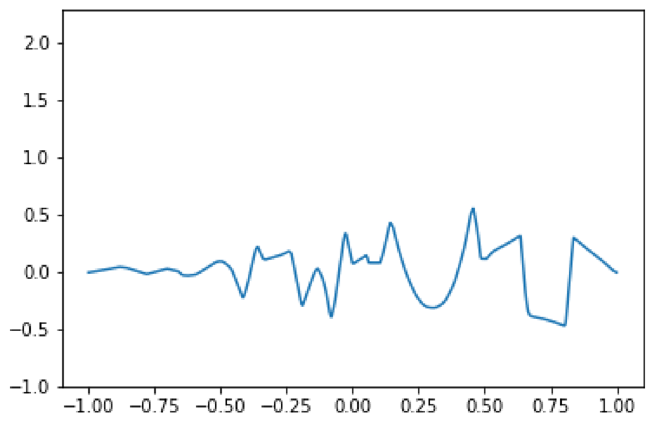


Fig. 30. Continued.

A



B



C

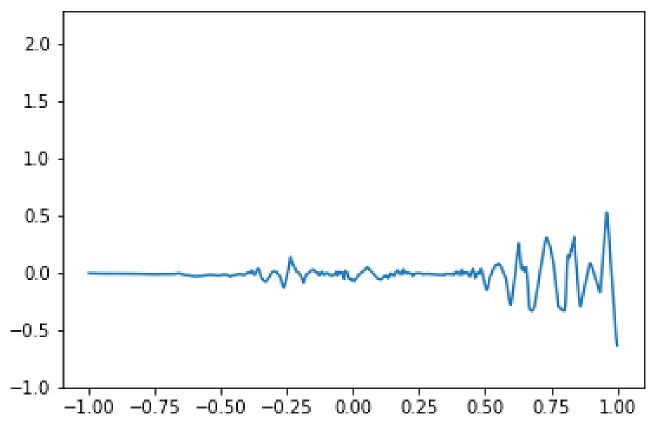
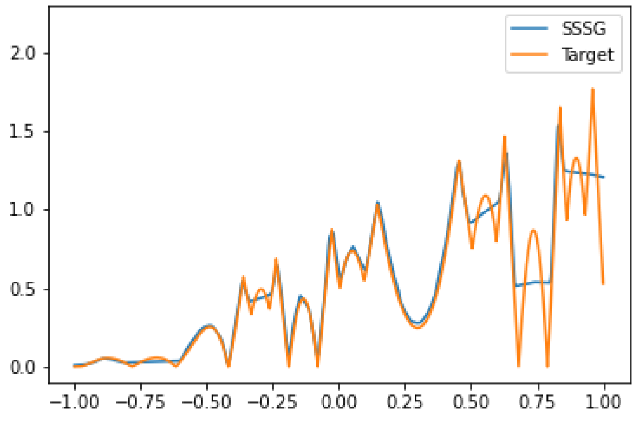
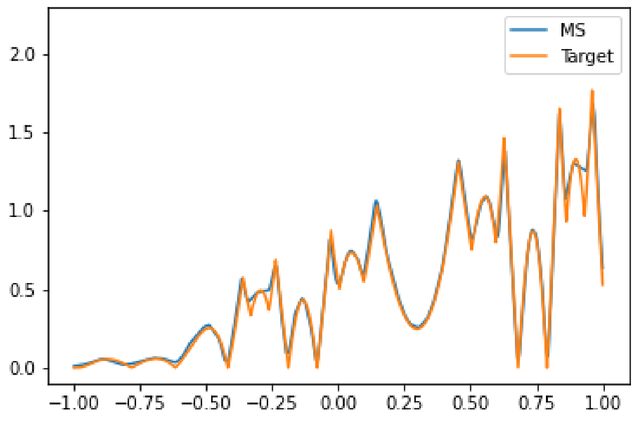


Fig. 31. For example 3 (noisy), a comparison of each grade in the multi-grade method, where each figure labeled A through C corresponds to grades one through three, respectively.

A



B



C

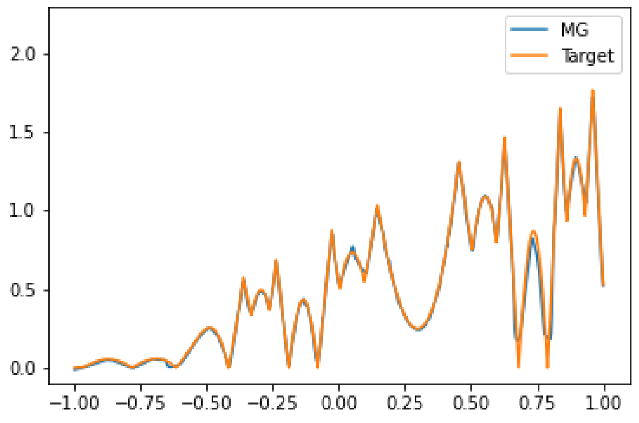


Fig. 32. For example 3 (noisy), the target function compared to the (A) single-scale, single grade model, (B) multi-scale model with eight scales, and (C) multi-grade model with three grades.

3.4 REMARKS

Observe from the numerical results presented in this section that for all examples (for both noise free and noisy cases), the multi-scale and multi-grade models outperforms the single-scale, single-grade model in many in terms of training time and accuracy. They are most resistant to noise. Moreover, multi-grade models provide a more customizable training regime.

CHAPTER 4

DEEPLY LEARNING DEEP INELASTIC SCATTERING KINEMATICS

Now, the multi-scale methods will be applied directly to a real world problem. The results presented here were part of a study conducted by the author [5].

Measurements of deep-inelastic scattering (DIS) by a multitude of experiments all-over the world [47, 48, 49, 50, 51] and the study of these measurements by the theoretical and experimental communities [51] have revealed information on the quark-gluon structure of nuclear matter and established quantum chromodynamics (QCD) as the theory of the strong interaction. The experiments at the HERA collider facility at the DESY research centre in Hamburg, Germany have played an important role in these studies. HERA has been the only electron-proton collider built so far [52]. The data collected in the years 1992 - 2007 have provided truly unique information on the internal structure of the proton and other hadrons [53].

The key component in these studies has been a precise reconstruction of the DIS kinematics, using information from the accelerator and the detectors. Multiple methods have been applied at the HERA experiments [54] to reach an optimal precision in each particular measurement. Each of the classical reconstruction methods uses only partial information from the DIS event and is subject to specific limitations, either arising from the detector or the assumptions used in the method.

With the DIS measurements at the upcoming Electron-Ion Collider in mind [55], a novel method using deep neural networks for the reconstruction of DIS kinematics based on supervised machine learning and study its performance using Monte Carlo simulated data from the ZEUS experiment [56] at HERA. the deep neural network models are optimised for the problem and are allowed to take full information from the DIS event into account. DNN models are trained on simulated data from the ZEUS experiment. The reconstruction is compared between the results from our trained model and the results from the classical reconstruction methods.

It is shown that the reconstruction of the DIS kinematics using deep neural networks provides a rigorous, data-driven method to combine and outperform classical reconstruction methods over a wide kinematic range. In the past, neural networks had already been used in the context of DIS experiments [57] and it is expected that this novel method and similar approaches will play an even more important role in ongoing and future DIS experiments [58, 59].

4.1 DEEP INELASTIC SCATTERING

Deep inelastic scattering is a process in which a high-energy lepton (l) scatters off a nucleon or nucleus target (h) with large momentum transfer (the momentum of each entity is given in parenthesis):

$$l(k) + h(P) \to l'(k') + \mathscr{H}(P') + \text{remnant}. \tag{4.1}$$

The detectors in collider experiments are designed to measure the final state of the DIS process, consisting of the scattered lepton l' and the hadronic final state (HFS) $\mathscr{H}$. The latter consists of hadrons with a relatively long lifetime as well as some photons and leptons but does not include the hadron remnant. The H1 [60] and ZEUS experiments were not able to register the remnant of the target due to its proximity to the proton beam pipe.

4.1.1 Deep Inelastic Scattering At Born Level

In the leading order (Born) approximation, the leptons interact with quarks in the hadrons by the exchange of a single virtual γ or Z boson in the neutral current (NC) reaction, and the exchange of single $W^{\pm}$ boson in the charged current (CC) reaction. The kinematics of the leading order DIS process in a Feynman diagram-like form is shown in Fig. 33.

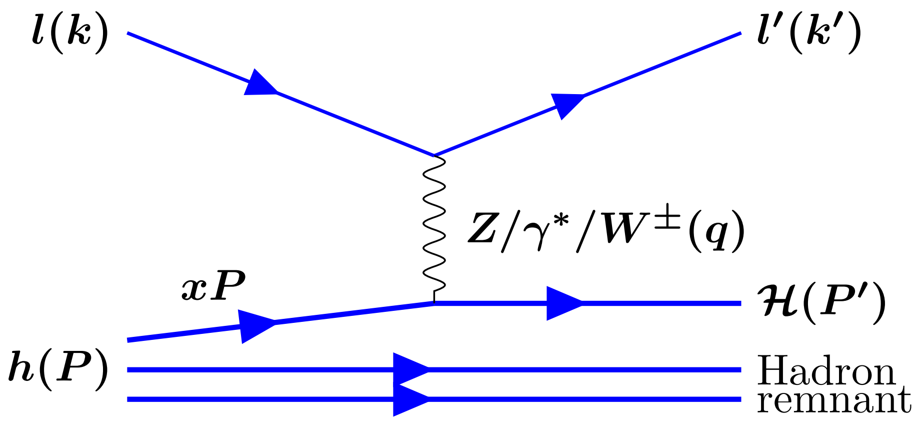


Fig. 33. Schematic representation of Deep Inelastic Scattering process at Born level.

In this paper, we will only consider the neutral current electron scattering off a proton in a collider experiment. In this reaction, the final state lepton is a charged particle (electron or positron) that can be easily registered and identified in the detector.

With a fixed centre-of-mass energy, $\sqrt{s} = \sqrt{(k+P)^2}$, two independent, Lorentz-invariant, scalar quantities are sufficient to describe the deep inelastic scattering event kinematics in the

Born approximation. Typically, the used quantities are:

- the negative squared four-momentum of the exchanged electroweak gauge boson:

$$Q^2 = -q \cdot q = -(k - k')^2, \tag{4.2}$$

- the Bjorken scaling variable, interpreted in the frame of a fast moving nucleon as the fraction of incoming nucleon longitudinal momentum carried by the struck parton:

$$x = \frac{Q^2}{2P \cdot q}. \tag{4.3}$$

In addition to that, the inelasticity y is used to define the kinematic region of interest. It is defined as the fraction of incoming electron energy taken by the exchanged boson in the proton rest frame

$$y = \frac{P \cdot q}{P \cdot k}. \tag{4.4}$$

Therefore, for the DIS an equation

$$Q^2 = syx \tag{4.5}$$

holds. However, the Born-level picture of the DIS process is not sufficient for the description of the observed physics phenomena. A realistic description of DIS requires the inclusion of higher order QED and QCD processes [61].

4.1.2 Higher Order Corrections to Deep-Inelastic Scattering Process

The DIS process with leading order QED corrections can be written as

$$l(k) + h(P) \to l'(k') + \gamma(P_\gamma) + X(P') + \text{remnant}, \tag{4.6}$$

so the kinematics is defined not only by the kinematic of the scattered electron and the struck parton, but also by the momentum of the radiated photon, P_γ. The lowest order electroweak radiative corrections can be depicted in a form of Feynman diagrams as shown in Fig. 34 A)-D) and should be considered together with the virtual corrections E)-G).

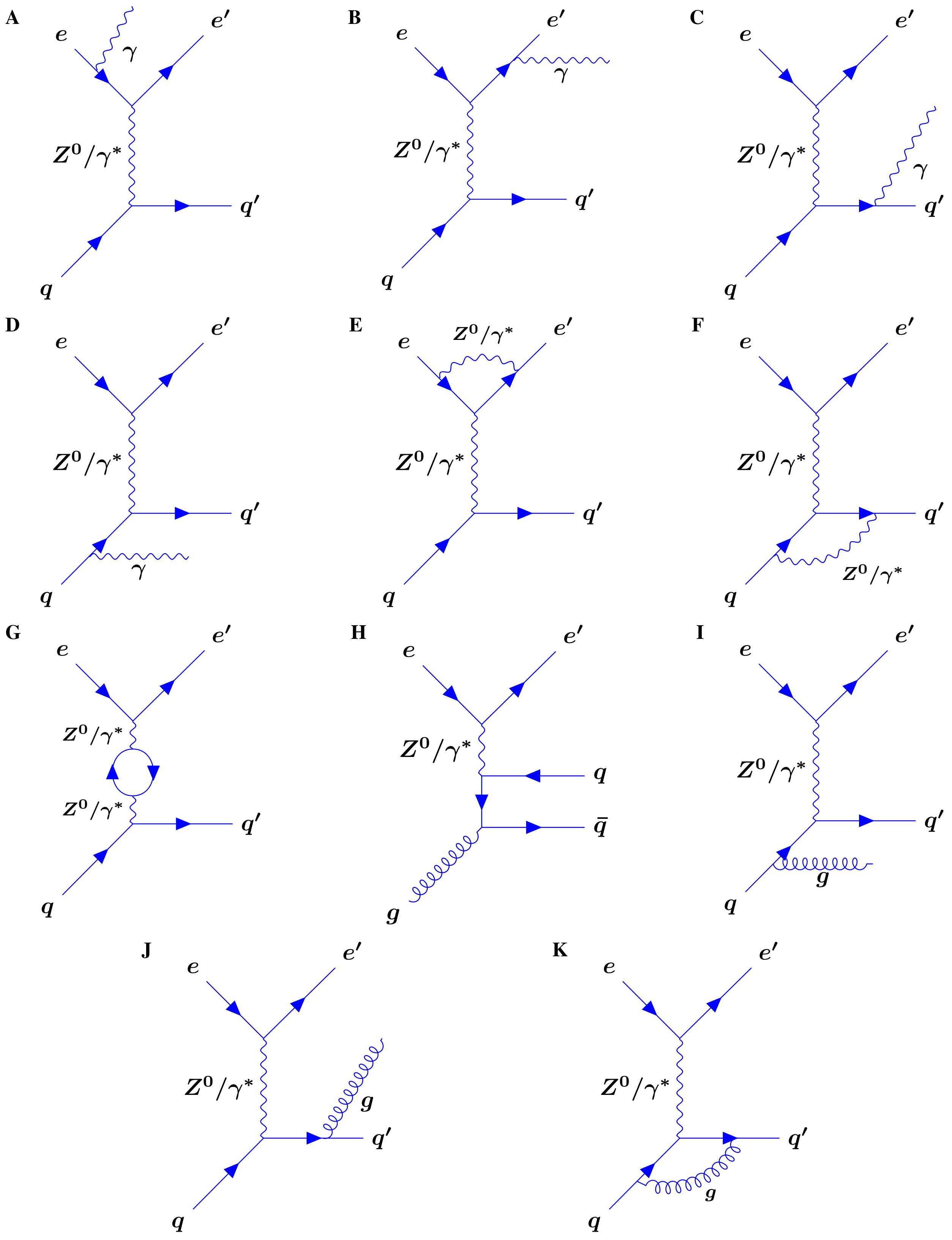


Fig. 34. Feynman diagrams for Deep Inelastic Scattering process with some leading order electroweak corrections A)-G) and QCD corrections H)-K). The proton remnant is omitted.

The Fig. 34 a),d) correspond to the initial state radiation (ISR) and the Fig. 34 b),c) to the final state radiation (FSR).

With the virtual corrections taken into account in the DIS process, Eq. (4.2) no longer holds, i.e.

$$q \cdot q \neq -(k-k')^2. \tag{4.7}$$

The presence of higher order QCD processes, (e.g. the boson-gluon fusion in Fig. 34 h) and QCD Compton Fig. 34 i),j)) makes the kinematic description of the DIS process even more complicated. Therefore, the exact definitions of the kinematic observables used in the analysis of the DIS events and the corresponding simulations are essential for the correct physics interpretation.

4.1.3 The Simulation of the DIS Events and the Kinematic Variables in the Simulated Events

The simulation of the inclusive DIS events in the Monte Carlo event generators (MCEGs) starts from the simulation at the parton level, i.e. the simulation of the hard scattering process and the kinematics of the involved partons, e.g. given by parton distribution functions (PDFs) [62] for the given hadron and considering processes with all types of partons in the initial state. The modelling of the hard scattering process combines the calculations of the perturbative QED and/or QCD matrix elements for the $2 \to n$ processes at parton level with the different QCD parton cascade algorithms designed to take into account at least some parts of the higher order perturbative QCD corrections not present in the calculations of matrix elements.

The simulated collision events on the particle (hadron) level are obtained using the parton level simulations as input and applying phenomenological hadronisation and particle decay models to them.

As of 2022, multiple MCEG programs are capable of simulating the inclusive DIS process at the hadron level with different levels of theory precision and sophistication of modelling of hadronisation, beam remnant and parton cascades, e.g. `Pythia6` [63], `Pythia8` [64], `SHERPA-MC` [65], `WHIZARD` [66] and `Herwig7` [67]. In addition to that, the `Lepto` [68], `Ariadne` [69], `Cascade` [70] and `Rapgap` [71] programs can simulate the DIS process using parts of the `Pythia6` framework for the simulation of hadronisation processes and decays of particles.

As it was discussed above, DIS beyond the Born approximation has a complicated structure which involves QCD and QED corrections [55]. The most recent MCEG programs, e.g. `Herwig7`, `Pythia8`, `SHERPA-MC`, or `WHIZARD`, contain these corrections as a particular case of their own general-purpose frameworks or are able to use specialised packages like `OpenLoops` [72], `blackhat` [73] or `MadGraph` [74]. In general, the modern MCEGs do not specify their definitions of the DIS kinematic observables, but in some cases they can be calculated from the kinematics of the initial and final state, both for the true and reconstructed kinematics. For instance, under some assumptions, the Q^2 in the event could be calculated according to Eq. (4.2). The total elimination of the ambiguities for such calculations is not possible, as the final state kinematics even at the parton level depend on the kinematics of all the emitted partons. The calculations of the kinematic observables from the momenta of particles at hadron level add an additional ambiguity related to the identification of the scattered lepton and the distinction of that lepton from the leptons produced in the hadronisation and decay processes.

Contrary to the approaches adopted in modern MCEG programs, the MCEGs used for the HERA experiments relied upon generator-by-generator implementations of the higher order QED and QCD corrections specific for DIS or alternatively applied `HERACLES` [61] for the corrections.

The way to get a simulation of the DIS collision even in a specific detector is the same as for any other type of particle collision event. It involves simulation of the particle transport through the detector material, simulation of detector response and is typically performed in `Geant` [75, 76] or similar tools. The simulated detector response is passed to the experiment-specific reconstruction programs and should be indistinguishable from the real data recorded by the detector and processed in the same way.

4.1.4 Reconstruction of the Kinematic Variables At the Detector Level

The kinematics of the DIS events are reconstructed in collider experiments by identifying and measuring the momentum of the scattered lepton l' and/or the measurements of the hadronic final state ($\mathscr{H}$). The identification of the scattered lepton is ambiguous even at the particle level of the simulated DIS collision events. The same ambiguity is present in the reconstructed real and simulated data at the detector level. Therefore, the identification of the scattered electron candidate for the purposes of physics analyses is a complicated task on itself and was a subject of multiple studies in the past, some of which also involved neural network-related techniques [57]. In our paper, we rely on the standard method of the electron identification at the ZEUS experiment [57] and discuss solely the reconstruction of the kinematic variables using the identified electron and other quantities measured in the detector.

The physics analyses performed in the experiments at HERA relied on the following quantities for the calculation of the kinematic observables x and Q^2:

- The energy ($E_{l'}$) and polar angle ($\theta_{l'}$) of the scattered electron. Most of the DIS experiments are equipped to register the scattered electron using the tracking and calorimeter detector subsystems. While the tracking system is able to provide information on the momentum of the scattered electron, the calorimeter system can be used to estimate the energy of the electron and the total energy of the collinear radiation emitted by the electron. At the detector level, the estimation of these energies can be done by comparing the momentum of the electron reconstructed in the tracking system and the energy deposits registered in the calorimeter system around the extrapolated path of the electron in the calorimeter.

- The energy of the HFS expressed in terms of the following convenient variables:

$$\delta_{\mathscr{H}} = \sum_{i \in \mathscr{H}} E_i - P_{Z,i} \tag{4.8}$$

and

$$P_{T,\mathscr{H}} = \sqrt{\left(\sum_{i \in \mathscr{H}} P_{X,i}\right) + \left(\sum_{i \in \mathscr{H}} P_{Y,i}\right)}, \tag{4.9}$$

where the sums run over the registered objects i excluding the scattered electron. Depending on the analysis requirements, the used objects could be either registered tracks, energy deposits in the calorimeter system, or a combination of both.

The measurements of the quantities listed above overconstrain the reconstruction of the DIS kinematics. Therefore, in the simplest case, any subset of two observables in $E_{l'}$, $\theta_{l'}$, $\delta_{\mathscr{H}}$, and $P_{T,\mathscr{H}}$, can be used for the reconstruction.

In our analysis, we consider three specific classical reconstruction methods based on these observables which were used by the ZEUS collaboration in the past: the electron, the Jacques-Blondel, and the double-angle methods. We briefly provide some details on the methods in this section, while a more detailed description can be found elsewhere [77].

The electron (EL) method uses only measurements of the scattered lepton, $E_{l'}$ and $\theta_{l'}$, to do the reconstruction of Q^2 and x. The kinematic variables calculated from these measurements are given by:

$$Q^2_{EL} = 2E_l E_{l'}(1+\cos\theta_{l'}) \tag{4.10}$$

and

$$x_{EL} = \frac{E_l E_{l'}(1+\cos\theta_{l'})}{E_P(2E_l - E_{l'}(1-\cos\theta_{l'}))}. \tag{4.11}$$

The electron method provides precise reconstruction of kinematics, but, since it uses only information from the scattered lepton, this method is affected by initial and final state QED radiation. Namely, the QED radiation registered in the detector separately from the scattered electron will not be taken into account in the calculations with this method. Practically, the reconstruction with this method gives reasonable results when E_l and $E_{l'}$ are significantly different from one another, but the resolution and stability becomes poor otherwise.

The Jacques-Blondel (JB) method uses only measurements of the final state hadronic system, $\delta_{\mathscr{H}}$ and $P_{T,\mathscr{H}}$, for the reconstruction. The kinematic variables are calculated from these by:

$$Q^2_{JB} = \frac{2E_l P^2_{T,\mathscr{H}}}{2E_l - \delta_{\mathscr{H}}}, \tag{4.12}$$

$$x_{JB} = \frac{2E_l Q^2_{JB}}{s\delta_{\mathscr{H}}}. \tag{4.13}$$

The JB method is resistant to possible biases because of unaccounted QED FSR, but requires precise measurements of the particles momenta in the hadronic final state. The factors that limit the precision of the measurements are the uncertainties in the particle identification, the finite resolution of the calorimeter and tracking detectors, the inefficiencies of these detectors, the impossibility of the particle detection around the beampipe, and the presence of objects that avoid detection (e.g. neutrinos from particle decays).

The double-angle (DA) method combines measurements from the scattered lepton and the final-state hadronic system, $\theta_{l'}$ and $\gamma_{\mathscr{H}}$, to perform the kinematic reconstruction as follows:

$$Q^2_{DA} = \frac{4E_l^2 \sin\gamma_{\mathscr{H}}(1+\cos\theta_{l'})}{\sin\gamma_{\mathscr{H}} + \sin\theta_{l'} - \sin(\gamma_{\mathscr{H}} + \theta_{l'})}, \tag{4.14}$$

$$x_{DA} = \frac{E_l \sin\gamma_{\mathscr{H}}(1+\cos\theta_{l'})}{E_P \sin\theta_{l'}(1-\cos\gamma_{\mathscr{H}})}, \tag{4.15}$$

where the angle $\gamma_{\mathscr{H}}$ is defined as

$$\cos\gamma_{\mathscr{H}} = \frac{P^2_{T,\mathscr{H}} - \delta^2_{\mathscr{H}}}{P^2_{T,\mathscr{H}} + \delta^2_{\mathscr{H}}}. \tag{4.16}$$

The angle $\gamma_{\mathscr{H}}$ depends on the ratio of the measured quantities $\delta_{\mathscr{H}}$ and $P_{T,\mathscr{H}}$, and thus, uncertainties in the hadronic energy measurement tend to cancel, leading to good stability of the reconstructed kinematic variables. Similar to the electron method, when E_l and $E_{l'}$ are significantly different from one another, the double-angle method provides reliable results, but the resolution and stability are poor otherwise.

4.1.5 The Methodology of Measurements in the Deep Inelastic *ep* Collisions

The methodology of measurements at lepton-hadron colliders in general is similar to the methodologies used at e^+e^- and hadron-hadron colliders. Briefly, in the most cases the quantity of interest is measured from the real data registered in the detector corrected for detector effects. The corrections are estimated by comparing the analysis at the detector level with the same analysis at particle level using detailed simulations of the collision events with the inclusion of higher order QED and QCD processes.

The main difference between the measurements at lepton-hadron colliders and elsewhere, is in the way the measurements involve collision kinematics. At e^+e^- colliders, the initial kinematics of the interactions is given by the lepton energies that are known parameters of the accelerators. Therefore, it is straightforward for most of the measurements in the e^+e^- experiments to estimate the centre-of-mass energy of the hard collision process. In hadron-hadron collider experiments, there is no way to measure the kinematic properties of the partons initiating the collision process, as the involved partons cannot be observed in a free state and most measurements in the hadron-hadron collisions are inclusive in the kinematics of the initial state. The DIS collisions at electron-proton colliders take a middle stance between these cases. The kinematic observables of the DIS process are measured on an event-by-event basis at the detector level using the methods described above.

In an experiment, the measurements of event kinematics is affected by various effects. For a proper comparison of the measurements of HFS, e.g. jet cross-sections or event shape observables to corresponding perturbative QCD (pQCD) predictions, the detector-level measurements are unfolded for detector effects while hadronisation correction factors are calculated using MCEGs or specialised programs and applied to the pQCD predictions [78, 79, 80, 81]. The prescription for calculation of those correction factors vary depending on the HFS quantities measured and the used definitions of the kinematic observables. Typically, at ZEUS and other experiments at HERA, after the unfolding of the detector effects, the measurements were also scaled by radiation correction factors to facilitate a comparison to theoretical calculations available at Born level in QED, see e.g. Ref. [82]. The factors were obtained from separate high-statistics MC simulations. This is a well-understood Monte Carlo approach and our deep learning technique can be used with it in exactly the same way as the classical methods, both for the experimental and the MC data. For future DIS measurements, e.g. at the upcoming Electron-Ion Collider, it is expected that the effects of QED and QCD radiation can be treated in an unified formalism [81], with the QED effects taken into account into a factorised approach. Our DNN-based reconstruction of DIS kinematics is compatible with such a factorised approach as well.

Therefore, in this analysis, we keep the calculation of correction factors for QED and hadronisation effects out of scope and limit the discussion to detector-level measurements. At particle-level in the MC generated events, we use a single definition of “true” kinematic observables that is based on the kinematics of the exchanged boson extracted from the MC event information.

Namely, we use the definitions of the true-level observables Q^2_{true} and x_{true} that follow the definitions implemented in ZEUS Common NTuples [83, 84] and were used in many ZEUS analyses. With these definitions the Q^2_{true} is calculated directly from the squared four-momentum of the exchange boson q_{boson},

$$Q^2_{\text{true}} = -|q_{\text{boson}}|^2. \tag{4.17}$$

The x_{true} is calculated according to the formula

$$x_{\text{true}} = \frac{Q^2_{\text{true}}}{y_{\text{after ISR}} s_{\text{after ISR}}}, \tag{4.18}$$

with

$$y_{\text{after ISR}} = \frac{q_{\text{boson}} \cdot P_{\text{proton beam}}}{(P_{\text{lepton beam}} - P_{\text{ISR radiation}}) \cdot P_{\text{proton beam}}} \tag{4.19}$$

and

$$s_{\text{after ISR}} = |P_{\text{proton beam}} + P_{\text{lepton beam}} - P_{\text{ISR radiation}}|^2, \tag{4.20}$$

where $P_{\text{proton beam}}$, $P_{\text{lepton beam}}$ and $P_{\text{ISR radiation}}$ represent the four-momenta of the proton beam particles, lepton beam particles and the momenta lost by the lepton beam particles to ISR. Thus, $y_{\text{after ISR}}$ corresponds to the fraction of energy of the bare lepton (i.e. without the ISR) transferred to the HFS in the centre of mass of the proton and $s_{\text{after ISR}}$ to the centre-of-mass-energy squared of the proton and the incoming bare electron.

4.2 MULTI-SCALE LEARNING METHODS

The reconstruction of DIS event kinematics is overconstrained by the previously mentioned measurements, $E_{l'}$, $\theta_{l'}$, $\delta_{\mathscr{H}}$, and $P_{T,\mathscr{H}}$. We trained an ensemble of multi-scale neural networks (NN) to reconstruct x and Q^2 by correcting results from classical reconstruction methods using the information on the scattered lepton and the final-state hadronic system.

The ensemble NN method presented here is a new approach designed specifically to address this reconstruction problem. The remainder of this section discusses in detail specifics of the NN architectures, the specific optimisation methods used to find the optimal parameters defining the DNNs, and the specific structure of the ensemble models. The main motivation for using the multi-scale model is:

- the universal approximation capabilities
- the necessary reduction in the approximation error with the increase in the depth
- avoidance of a degradation problem [85] in training due to the residual structure of the models.

We use a logarithmic loss function. The specifics of the model follow. However, due to the large size of the models, to avoid overfitting, we use ℓ^1 regularization.

We use the stochastic gradient descent algorithm with momentum for the optimizer to solve for the optimal parameters in the optimization problem.

We construct a model to rigorously weight classically derived reconstructions of x and Q^2 with corrections based on measurements from the final state lepton and hadronic system. The final reconstruction of these observables with the neural network approach are labelled below as Q^2_{NN} and x_{NN} respectively.

The constructed model is an ensemble of networks from the previously defined function class multi-scale deep neural networks, where the activation function σ is the rectified linear unit **ReLU**. The values of the depth and width of each hidden layer varies with each network in the ensemble.

The NC DIS events studied in this analysis are by definition the events containing the scattered electron in the final state, therefore we aim to reconstruct the Q^2_{NN} primarily from the related observables, i.e. using the properties of the electron directly measured in the experiment. In particular, we use as inputs three set of variables: the classically reconstructed kinematic observables $\left(Q^2_{EL}, Q^2_{DA}, Q^2_{JB}\right)$, measurements on the scattered lepton $(E_{l'}, \theta_{l'})$, and measurements on the final-state hadronic system $\left(\delta_{\mathscr{H}}, P_{T,\mathscr{H}}\right)$. We reconstruct the Q^2 in the form:

$$\begin{aligned} Q^2_{NN} = {} & A_{Q^2}\left(Q^2_{EL}, Q^2_{DA}, Q^2_{JB}\right) + L_{Q^2}\left(A_{Q^2}, E_{l'}, \theta_{l'}\right) + \\ & + H_{Q^2}\left(A_{Q^2}, \delta_{\mathscr{H}}, P_{T,\mathscr{H}}\right), \end{aligned} \tag{4.21}$$

in which A_{Q^2} could be understood as a rigorous average of classically derived reconstructions, L_{Q^2} is a correction term computed from the scattered lepton, and H_{Q^2} is another correction term computed from the final-state hadronic system. In our analysis A_{Q^2}, L_{Q^2}, H_{Q^2} are simultaneously trained multi-scale networks in mapping three inputs to one output, with 5 hidden layers. Each hidden layer of the networks contains 2000 nodes.

The x observable for the NC DIS events is actually calculated from the electron-related observables as well. Therefore, we reconstruct x_{NN}, with Q^2_{NN} also as an input, for a total of eight inputs $(x_{EL}, x_{DA}, x_{JB}, E_{l'}, \theta_{l'}, \delta_{\mathscr{H}}, P_{T,\mathscr{H}}, Q^2_{NN})$, in the form:

$$\begin{aligned} x_{NN} = {} & A_x\left(x_{EL}, x_{DA}, x_{JB}\right) + L_x\left(A_x, Q^2_{NN}, E_{l'}, \theta_{l'}\right) + \\ & + H_x\left(A_x, Q^2_{NN}, \delta_{\mathscr{H}}, P_{T,\mathscr{H}}\right), \end{aligned} \tag{4.22}$$

where A_x, L_x, and H_x are defined similarly to A_{Q^2}, L_{Q^2}, and H_{Q^2}, but A_x is a multi-scale network with 20 hidden layers, where each hidden layer contains 1000 nodes, and L_x, H_x are multi-scale networks with 10 hidden layers, where each hidden layer contains 500 nodes.

The number of hidden layers and the number of nodes in each hidden layer were each progressively increased until sufficiently desirable results were achieved. Smaller networks provided good results on average, but larger networks were needed to find best results in small, specific regions of the kinematic space. A further increasing of the numbers of hidden layers and nodes per layer beyond the chosen values was found to not significantly alter the performance of the kinematic reconstruction, due to the convergence results of deep neural networks. The convergence theorems of deep neural networks with the ReLU activation function as the number of layers increases were recently established in [26, 86].

In the ensemble neural network model, we used the **ReLU** function as the nonlinear activation function.

It has been shown [87] that with a gradient descent algorithm, using the **ReLU** function as the activation function provides a smaller training time compared to that with the use of functions with saturating nonlinearities, such as a sigmoid or hyperbolic tangent function. The reduced training time enabled us to experiment with more sophisticated networks. In addition to this, the **ReLU** functions do not need input normalisation to prevent them from saturating, which is a desirable property for the present analysis.

Moreover, it was shown in [26] that the selection of the **ReLU** activation function produces a neural network as a piecewise linear function over a nonuniform partition, of the domain. Such a structure ensures a good representability of the neural network and can overcome the problem of underfitting.

To accommodate for the large range of the Q^2 and x variables in the analysis, we select the loss function defined in Eq. (2.37) for the training of the NN models.

4.3 EXPERIMENTAL SETUP

The Monte Carlo simulated events used to train our deep neural networks were specifically generated using the conditions of $e^{\pm}p$ scattering in the ZEUS detector at HERA. A detailed description of the ZEUS detector can be found elsewhere [56].

In our analysis, we have used two samples of Monte Carlo simulated $e^{+}p$ DIS events that are provided by the ZEUS collaboration. These samples were generated with an inclusion of QED and higher order QCD radiative effects using the `HERACLES 4.6.6` [61] package with `DJANGOH 1.6` [88] interface and the `ARIADNE 4.12` and `LEPTO 6.5.1` packages for the simulation of the parton cascade. For both samples the same set of kinematic cuts was applied during the generation, the same set of PDFs were used, CTEQ5D [89] and the same hadronisation settings were used to model the hadronisation with the `Pythia6` program. Therefore, the essential difference between the two samples is the way the higher order corrections are partially modelled with the corresponding algorithms (QCD cascades). Namely, the `LEPTO` MCEG utilises the parton shower approach [90], while `ARIADNE` implements a color-dipole model [91]. Accordingly, we label the data-sets produced by the `LEPTO` generator as "CDM" data sets and those with `ARIADNE` as "MEPS" data sets.

The generated particle-level events were passed through the ZEUS detector and trigger simulation programs based on `Geant` 3.21 [75], assuming the running conditions of the ZEUS experiment in the year 2007 with a proton beam energy of 920 GeV. The simulated detector response was processed and reconstructed using exactly the same software chain and the same procedures as being used for real data. The results of the processing were saved in ROOT [92] files in a form of ZEUS Common NTuples [83, 84], a format that can be easily used for physics analysis without any ZEUS-specific software.

4.4 EVENT SELECTION

The selection of events for the neural network training is motivated by the selection procedure applied in the previous ZEUS analyses [93, 94, 95, 96, 97, 98]. Even though the presented analysis performed on the Monte Carlo simulated events, the selection cuts are choosen and applied as if the analysis is performed on real data for the purpose of being as close as possible to the measurements. The general motivation for these cuts is the same as in many analyses performed by the ZEUS collaboration: to define unambiguously the phase space of the measurement, ensure low fraction of background events, and a reasonable description of the detector acceptance by Monte Carlo simulations.

4.4.1 Phase Space Selection

The phase space for the training of the neural networks in this analysis is selected as $100\,\text{GeV}^2 < Q^2 < 20480\,\text{GeV}^2$, being close to the phase space of the physics analysis in Ref. [97].

In the phase-space region at low x and very low inelasticity y, the QED predictions from the Monte Carlo simulations are not reliable because of a limit of higher orders in the calculations [99]. To avoid these phase-space regions, the events are required to have $y_{JB} \cdot (1 - x_{DA})^2 > 0.004$ and $y_{JB} > 0.04$ [99]. To ensure optimal electron identification and electron energy resolution, similar to the previous physics analyses, a kinematic cut $0.2 < y_{EL} < 0.7$ is used.

4.4.2 Event Selection

The deep inelastic scattering events of interest are those characterised by the presence of a scattered electron in the final state and a significant deposit of energy in the calorimeter from the hadronic final state. The scattered electrons are registered in the detector as localised energy depositions primarily in the electromagnetic part of the calorimeter, with little energy flow into the hadronic part of the calorimeter. On the other hand, hadronic showers propagate in the detector much more extensively, both transversely and longitudinally.

In addition to the DIS process, there are also background processes which leave similar signatures in the detector as those described above. Therefore, the correct and efficient identification of the scattered lepton is crucial for the selection of the NC DIS events. For this analysis, the `SINISTRA` algorithm is used to identify lepton candidates [57]. Based on the information from the detector and the results of the `SINISTRA` algorithm, the following selection criteria are applied to select the events for the further analysis:

- **Detector status:** It is required that for all the events the detector was fully functional.
- **Electron energy:** At least one electron candidate with energy greater than 10 GeV [97] is identified in the event.
- **Electron identification probability:** The `SINISTRA` [57] probability of a lepton candidate being the DIS lepton was required to be greater than 90%. If several lepton candidates satisfy this condition, the candidate with the highest probability is used. In addition to this, there must be no problems reported by the `SINISTRA` algorithm.
- **Electron isolation:** To assist in removing events where the energy deposits from the hadron system overlap with those of the scattered lepton, the fraction of the energy not associated to the lepton is required to be less than 10% over the total energy deposited within a cone around the lepton candidate. The cone is defined with a radius of 0.7 units in the pseudorapidity-azimuth plane around the lepton momentum direction [97].
- **Electron track matching:** The tracking system covers the region of polar angles restricted to $0.3 < \theta < 2.85$ rad. Electromagnetic clusters within that region that have no matching track are most likely photons. If the lepton candidate is within this region, the presence of a matched track is required. This track must have a distance of closest approach between the track extrapolation point at the front surface of the CAL and the cluster centre-of-gravity-position of less than 10 cm. The track energy must be greater than 3 GeV [97].
- **Electron position:** To minimise the impact of imperfect simulation of some detector regions, additional requirements on the position of the electromagnetic shower are imposed.

The events in which the lepton is found in the following regions (x,y and z being the cartesian coordinates in the ZEUS detector) are rejected [100]:

- $\sqrt{x^2+y^2} < 18$ cm, regions close the beam pipe
- $z < -148$ cm and $y > 90$ cm and $-14 < x < 12$ cm, a part of the RCAL where the depth was reduced due to the cooling pipe for the solenoid (chimney),
- $-104 < z < -98.5$ cm or $164 < z < 174$ cm, regions in-between calorimeter sections (super-crack).

- **Primary vertex position:** It was required that the reconstructed primary vertex position is close to the central region of the detector, applying the selection $-28.5 < Z_{\text{vtx}} < 26.7$ cm [97].

- **Energy-longitudinal momentum balance:** To suppress photoproduction and beam-gas interaction background events and imperfect Monte Carlo simulations of those, restrictions are put on the energy-longitudinal momentum balance. This quantity is defined as:

$$\delta = \delta_l + \delta_{\mathscr{H}} = (E_{l'} - P_{z,l'}) + (E_{\mathscr{H}} - P_{z,\mathscr{H}}) = \sum_i (E_i - P_{z,i}), \tag{4.23}$$

  where the final summation index runs over all energy deposits in the detector. In this analysis, we apply a condition of $38 < \delta < 65\,\text{GeV}$ [97].

- **Missing transverse energy:** To remove beam-related background and cosmic-ray events, a cut on the missing energy is imposed. $P_{T,\text{miss}}/\sqrt{E_T} < 2.5\,\text{GeV}^{1/2}$, where E_T is the total transverse energy in the CAL and $P_{T,\text{miss}}$ is the missing transverse momentum, the transverse component of the vector sum of the hadronic final state and scattered electron momenta.

4.5 TRAINING THE NN MODELS

We use the neural network models defined in Sec. 4.2 and consider them as optimisation problem in terms of Eq. (2.37), i.e., we minimise the loss function across the selected training set and satisfy sparse regularity conditions. Every neural network making the ensemble model for x and Q^2 are trained simultaneously. The optimal regularity condition depends on the selection of the training set, the events batch size, the initial learning rate, and the regularisation parameter. We aim to select these in a way that minimises overfitting in particular regions of the kinematic space while still maximising the mean accuracy of the model.

We select a set of events from the "MEPS" data sets for training, as described in Sec. 4.4, and define the true values of the x and Q^2 as described in Sec. 4.1.3. Fig. 35 shows a distribution of selected events in the $(x_{\text{true}}, Q^2_{\text{true}})$ plane and the boundaries of the chosen analysis bins.

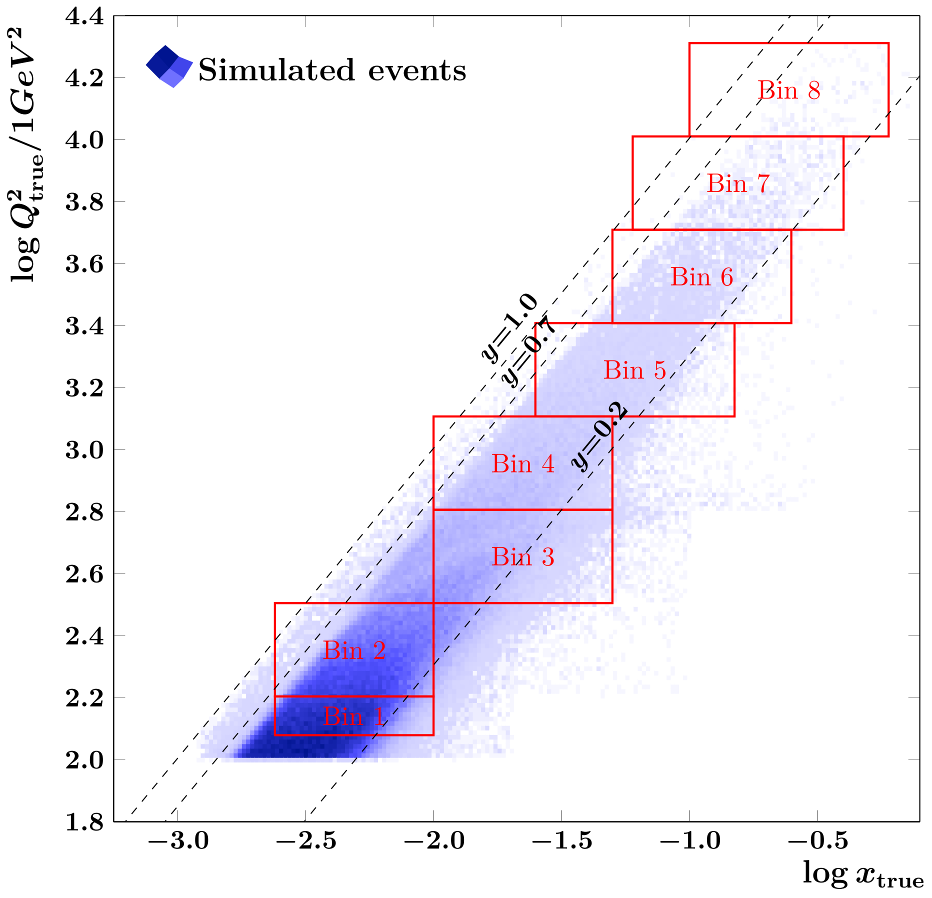


Fig. 35. Distribution of events from the training set in $(x_{\mathrm{true}}, Q^2_{\mathrm{true}})$ plane and the boundaries of the analysis bins from Tab. 21.

First, we train the network to reconstruct Q^2 by optimizing Eq. (2.37) with an initial learning rate of $L = 1.0 \times 10^{-5}$ and a regularisation parameter of $R = 1.0 \times 10^{-5}$. We select a batch size of 10000.

The reconstruction of x is more complex. We fix the regularization parameter to $R = 1.0 \times 10^{-5}$ and select optimal parameters for the learning rate L and batch size B experimentally by varying the learning rate against the batch size. The appropriate selection of these parameters ensures fast convergence of the stochastic gradient method by balancing noise in the gradients with the stability of the algorithm. For each set of parameters, ten attempts are made and the best result in terms of the mean square error of the x reconstruction model over the training set is chosen. The results are listed in Tab. 19. The smaller learning rate assures a higher stability of the results than a larger learning rate. It prolongs the training process but avoids a poor convergence of the learning process as observed for larger learning rates. The larger batch size does not offer any advantages in our analysis as shown in Tab. 19. We select an initial learning rate of 10^{-5} with a minimal batch size.

TABLE 19
Resolution of $\log x$ reconstruction after 200 epochs of training with different values of initial learning rate $\mathscr{L}$ and batch size $\mathscr{B}$.

	RMS of $\log x - \log x_{\text{true}}$		
$\mathscr{L}$	$\mathscr{B}$ 10000	$\mathscr{B}$ 50000	$\mathscr{B}$ 100000
1.0×10^{-7}	0.1507	0.1523	0.1598
5.0×10^{-7}	0.1568	0.1575	0.1575
1.0×10^{-6}	0.1504	0.1585	0.1547
5.0×10^{-6}	0.2384	0.2284	0.1829
1.0×10^{-5}	0.2182	2.5972	2.2767
1.5×10^{-5}	0.2122	1.7751	1.6028
2.0×10^{-5}	3.0258	0.2607	0.2852
5.0×10^{-5}	3.6962	2.4858	3.1231

To choose the regularization parameter close to the optimal one, we vary its value with constant batch size of 10000 and initial learning rate of 10^{-5} and again observe the mean square error of the x reconstruction model over the training set. For each set of parameters, ten attempts are made and the best result is chosen. The results are presented in Tab. 20. Accordingly, we choose a regularization parameter of 10^{-6}. Using this regularization, the neural network models for both x and Q^2 are defined by weight parameters, of which 50% are effectively zero, or less than 10^{-8}

Following the suggestions in Ref. [101], we start with a small batch size, and increase it in initial training epochs.

We test this approach by comparing the mean square error of the x reconstruction model over the training set over the first 200 epochs of training over three different training regimes. The results are summarised in Fig. 36 and imply to use a gradually increasing batch size up to a maximum batch size of 1000.

4.6 RESULTS

We evaluate the performance of our approach for the reconstruction of DIS kinematics by applying it to detailed Monte Carlo simulations from the ZEUS experiment and by comparing our results to the results from the electron, double-angle, and Jacques-Blondel reconstruction methods.

TABLE 20

Resolution of $\log x$ reconstruction after 200 epochs of training with different values of regularisation parameter $\mathscr{R}$.

$\mathscr{R}$	RMS of $\log x - \log x_{\text{true}}$
1.0×10^{-6}	0.1493
1.0×10^{-5}	0.1494
1.0×10^{-4}	0.1484

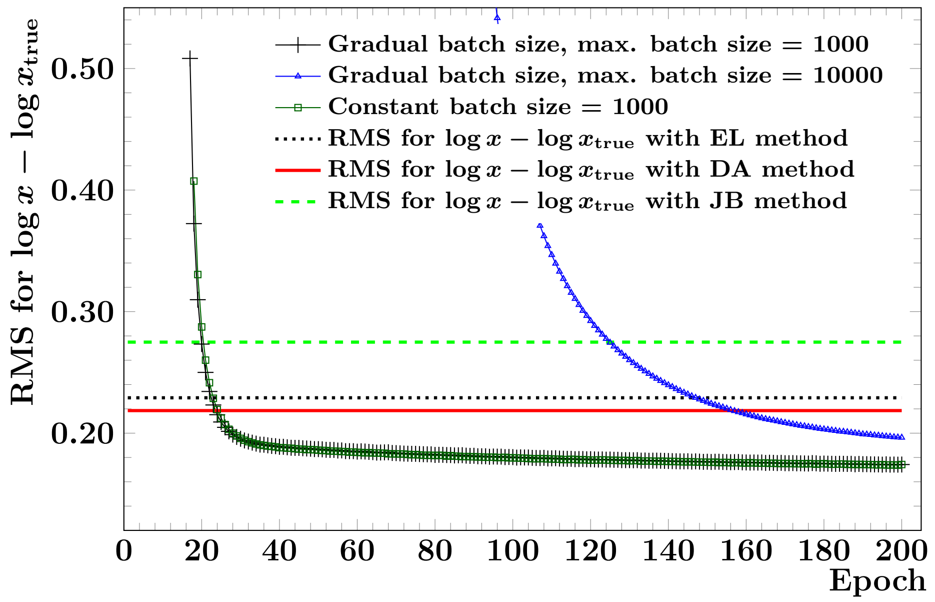


Fig. 36. Training history for x reconstruction model using different training parameters but the same Q^2 reconstruction model obtained with $L = 1.0 \times 10^{-5}$, $R = 1.0 \times 10^{-6}$ and $B = 10000$. In each of the cases, the initial learning rate was set to $L = 1.0 \times 10^{-5}$ and the regularisation parameter to $R = 1.0 \times 10^{-6}$.

TABLE 21

Kinematic bins in x and Q^2, see also Fig. 35.

Bin	Q^2 (GeV^2)	x
1	120 - 160	0.0024 - 0.010
2	160 - 320	0.0024 - 0.010
3	320 - 640	0.01 - 0.05
4	640 - 1280	0.01 - 0.05
5	1280 - 2560	0.025 - 0.150
6	2560 - 5120	0.05 - 0.25
7	5120 - 10240	0.06 - 0.40
8	10240 - 20480	0.10 - 0.60

For the comparison, we use various combinations of statistically independent data sets, one for the training, and another for the evaluation. In our systematic studies, we have found no signs of overtraining and also no indication that the results depend on the selected Monte Carlo simulations. For the results presented in this section, we use the “MEPS” data set for the training and the “CDM” data set for evaluation. The main quantities of the comparison are the resolutions of the reconstructed variables $\log Q^2/1\,\mathrm{GeV}^2$ and $\log x$ as measured in selected $x - Q^2$ regions (bins). The resolutions are defined as

$$\sqrt{\sum_i^N \left(\log Q_i^2/1\,\mathrm{GeV}^2 - \log Q_{i,\mathrm{true}}^2/1\,\mathrm{GeV}^2\right)^2/N}$$

and

$$\sqrt{\sum_i^N (\log x_i - \log x_{i,\mathrm{true}})^2/N},$$

where N stands for the number events in the corresponding bin. The boundaries of the bins are given in Tab. 21 and are chosen to be close to the bins used in ZEUS DIS analyses, e.g. in Ref. [95].

The distributions of the $\log Q^2/1\,\mathrm{GeV}^2 - \log Q_{\mathrm{true}}^2/1\,\mathrm{GeV}^2$ and $\log x - \log x_{\mathrm{true}}$ quantities are given in the Fig. 37 and Fig. 38, respectively. The numerical values for the resolution are summarised for all the bins and methods in Tab. 22. The NN optimization procedure minimises the generalization error described in Eq. (2.37) plus the regularization penalty, so distributions in Figs. 37 and 38 do not necessarily peak at zero. In addition to that, Fig. 39 and Fig. 40 show the two dimensional distributions of events in $\log Q^2/1\,\mathrm{GeV}^2$ vs. $\log Q^2_{\mathrm{true}}/1\,\mathrm{GeV}^2$ and $\log x$ vs. $\log x_{\mathrm{true}}$ planes.

The comparison of the NN-based approach with the classical methods demonstrates that the NN-based approach is well suited for the reconstruction of DIS kinematics. Specifically, for most of the bins, our approach provides the best resolution as measured by the standard deviation of the logarithmic differences of true and reconstructed variables. The better performance of the NN-based approach in most of the bins is a consequence of using additional available information about the final state. In this respect, the NN-based approach is similar to averaging of the values provided by the classical methods with some weights or to alternative approaches for the same task, e.g. see kinematic fitting in Ref. [102]. However, in addition to a better resolution, the reconstruction with the NN has an important advantage over the classical methods or any simple combination of them. It allows to combine the methods without the intrinsic biases of each method and enables an extension of the model with additional physics observables in a robust way.

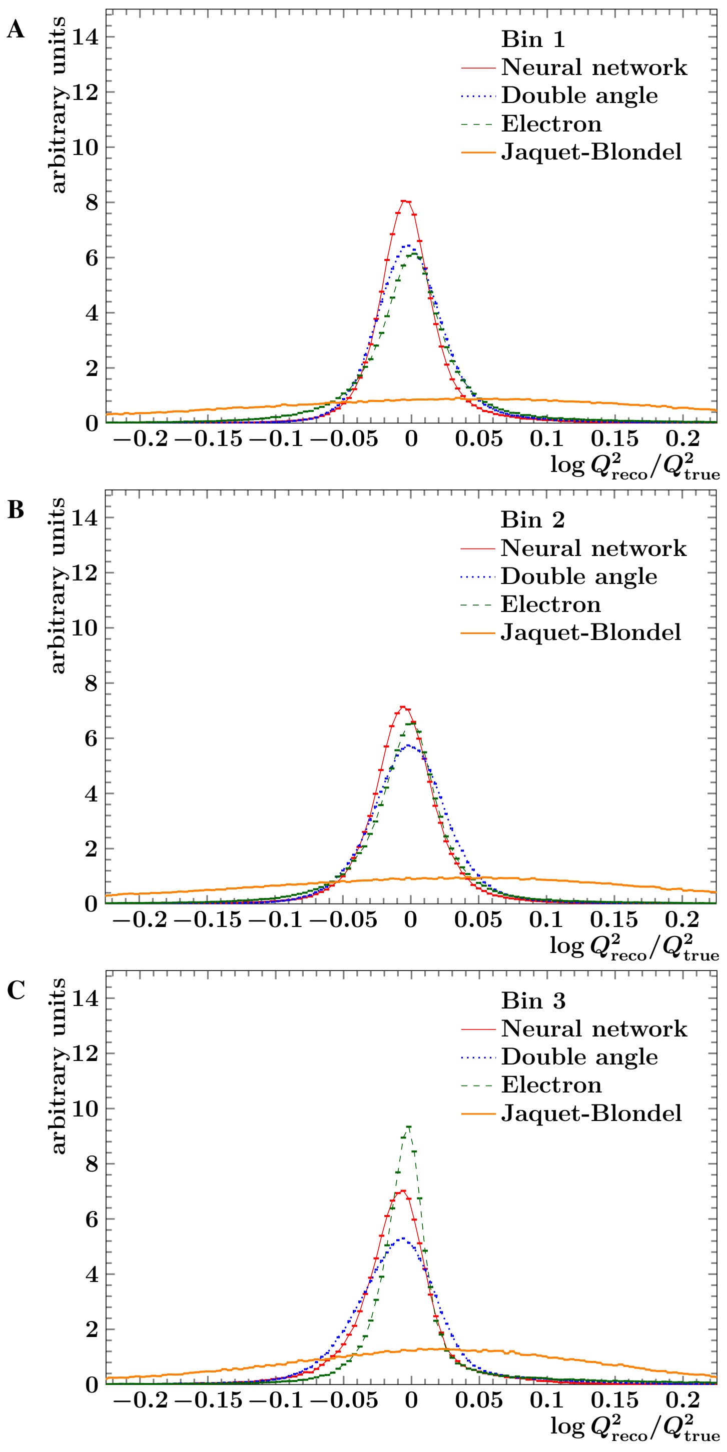
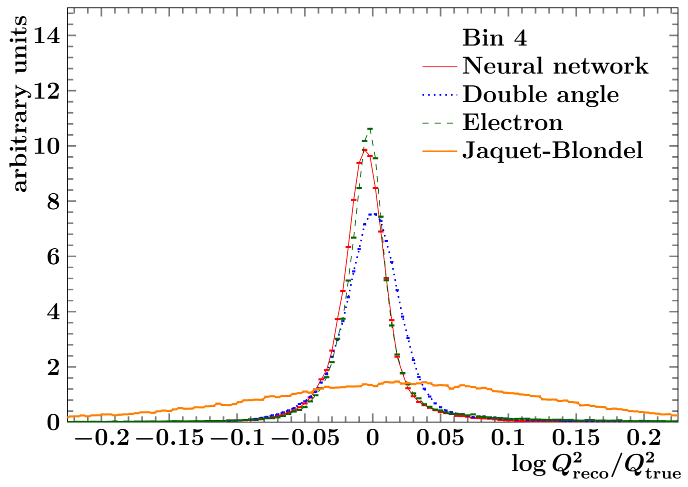
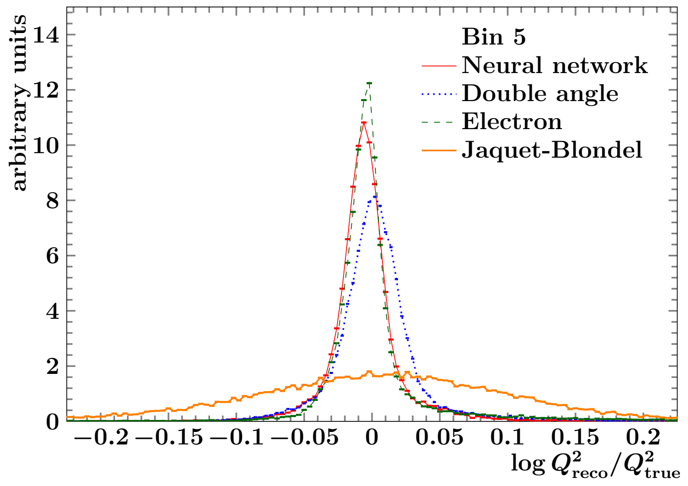


Fig. 37. Distributions of $\log Q^2/1\,\mathrm{GeV}^2 - \log Q^2_{\mathrm{true}}/1\,\mathrm{GeV}^2$ for various reconstruction methods in individual analysis bins. For better visibility, the data points for each reconstruction method are connected with straight lines.

D



E



F

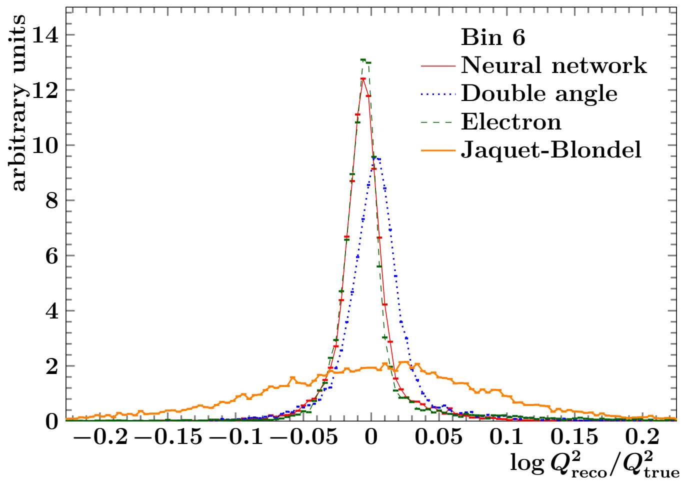
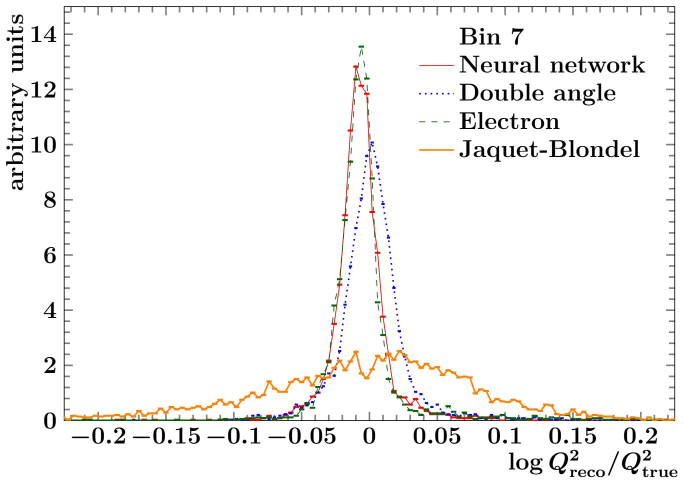


Fig. 37. Continued.

G



H

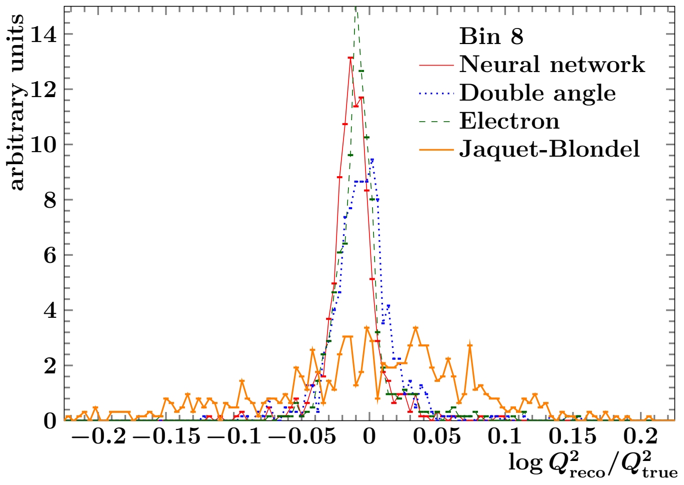


Fig. 37. Continued.

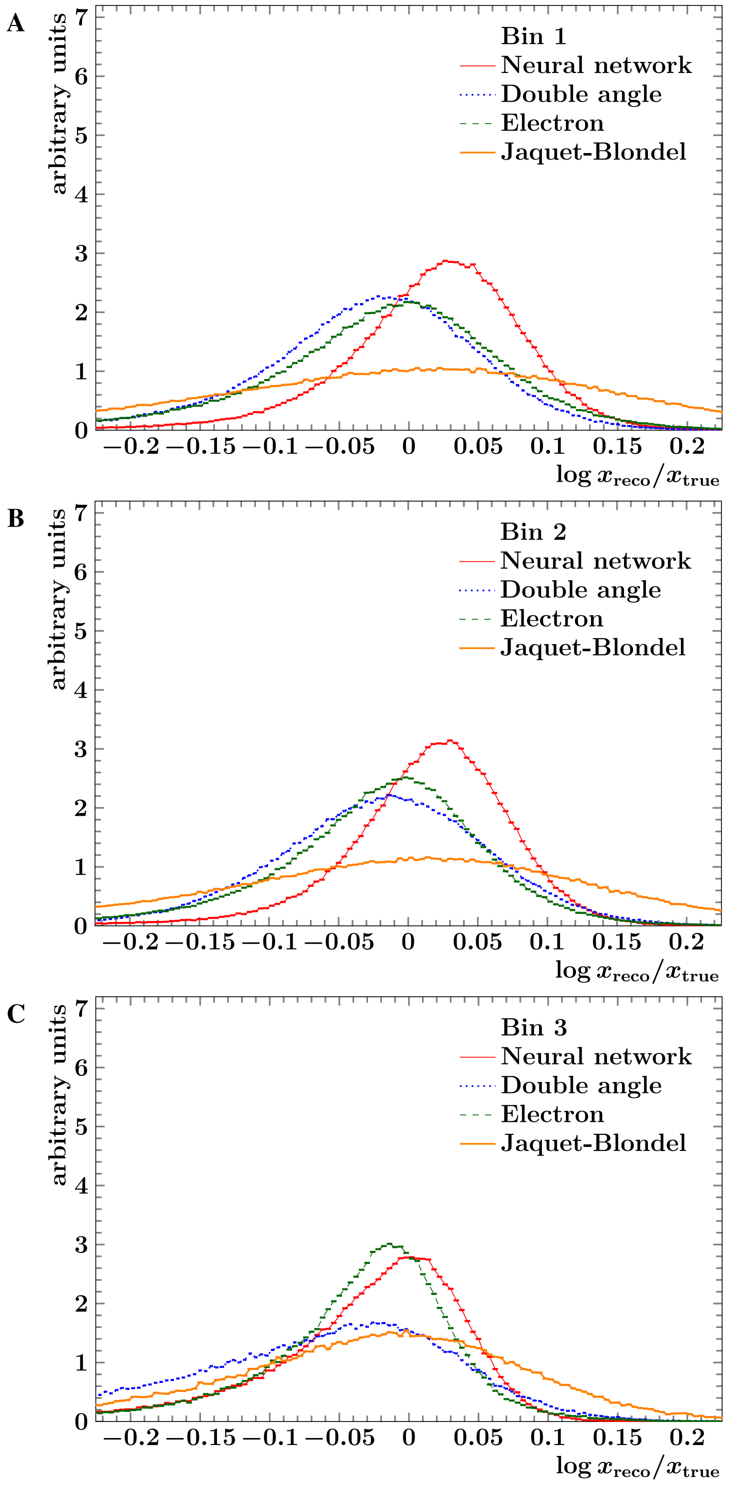


Fig. 38. Distributions of $\log x - \log x_{\mathrm{true}}$ for various reconstruction methods in individual analysis bins. For better visibility, the data points for each reconstruction method are connected with straight lines.

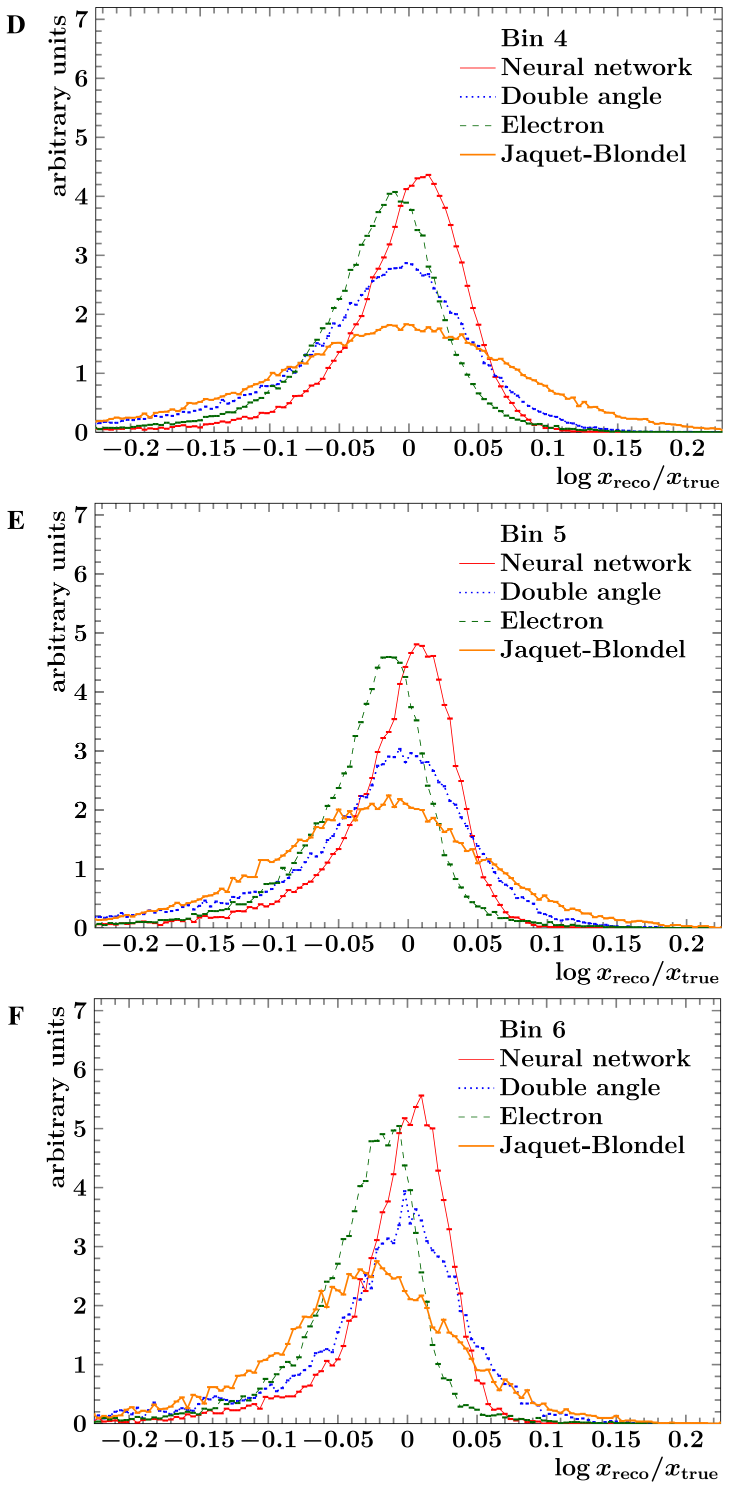
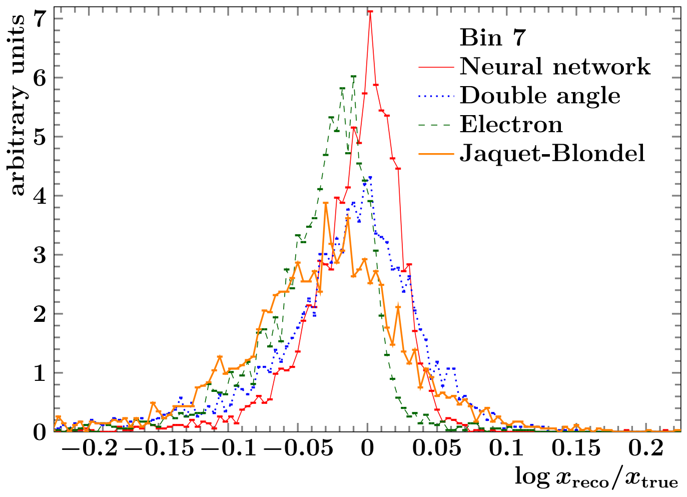


Fig. 38. Continued.

G


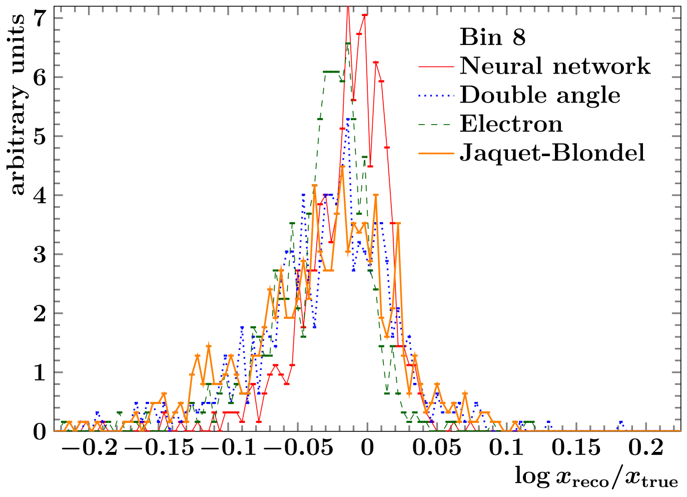
H


Fig. 38. Continued.

TABLE 22

Resolution of the reconstructed kinematic variables in bins of x and Q^2. The resolution for x and Q^2 is defined as the RMS of the distributions $\log(x) - \log(x_{\text{true}})$ and $\log(Q^2) - \log(Q^2_{\text{true}})$ respectively.

Bin	Events	Resolution of		Resolution of	
		$\log x$, $\times 10^3$		$\log Q^2/1\,\text{GeV}^2$, $\times 10^3$	
1	301780	**NN: 70**	EL: 83	**NN: 35**	EL: 35
		JB: 180	DA: 103	JB: 203	DA: 62
2	350530	**NN: 69**	EL: 82	**NN: 40**	EL: 43
		JB: 167	DA: 96	JB: 192	DA: 64
3	138456	**NN: 98**	EL: 130	NN: 55	**EL: 53**
		JB: 138	DA: 100	JB: 150	DA: 77
4	74844	**NN: 67**	EL: 84	**NN: 44**	EL: 46
		JB: 117	DA: 77	JB: 138	DA: 63
5	31043	**NN: 64**	EL: 91	**NN: 36**	EL: 41
		JB: 102	DA: 73	JB: 117	DA: 53
6	11475	**NN: 53**	EL: 79	**NN: 33**	EL: 36
		JB: 83	DA: 61	JB: 100	DA: 45
7	3454	**NN: 50**	EL: 69	**NN: 36**	EL: 38
		JB: 74	DA: 55	JB: 93	DA: 42
8	624	**NN: 36**	EL: 55	**NN: 33**	EL: 37
		JB: 67	DA: 45	JB: 95	DA: 41

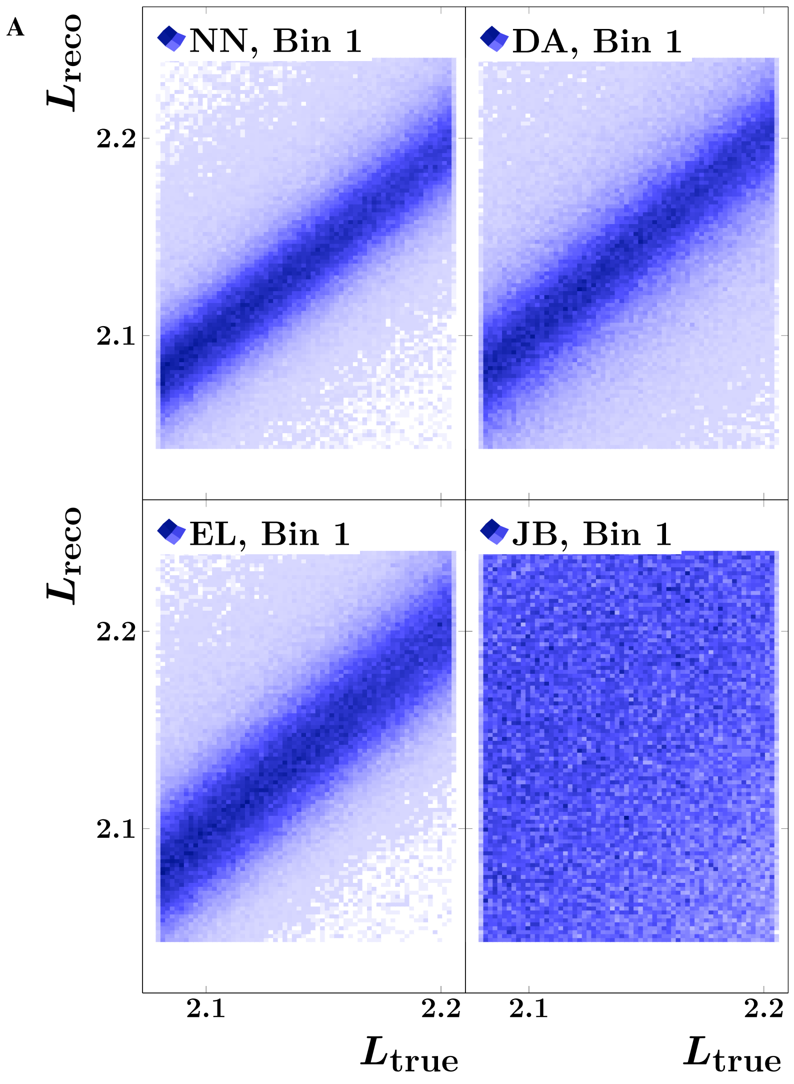


Fig. 39. Distribution of events in $L_{\text{reco}} = \log Q^2/1\,\text{GeV}^2$ versus $L_{\text{true}} = \log Q^2_{\text{true}}/1\,\text{GeV}^2$ plane for different reconstruction methods in individual analysis bins.

The resolution is improved by the NN-based approach in two ways. For the first couple of bins, the main improvement is caused by the more precise estimation of the reconstructed observables. This is clearly seen in Figs. 37 and 38. For the bins with higher Q^2 and x the main improvement is due to the rejection of outliers, which can be seen in Figs. 39 and 40. The bins with higher Q^2 and x also demonstrate another, very specific advantage on the DNN approach. Due to the low number of training events in the high Q^2 and x region, it would not be possible to train a DNN model or combine the Q^2 and x observables with other methods using information from this region only. However, the DNN training process benefits from the constraints from the higher number of

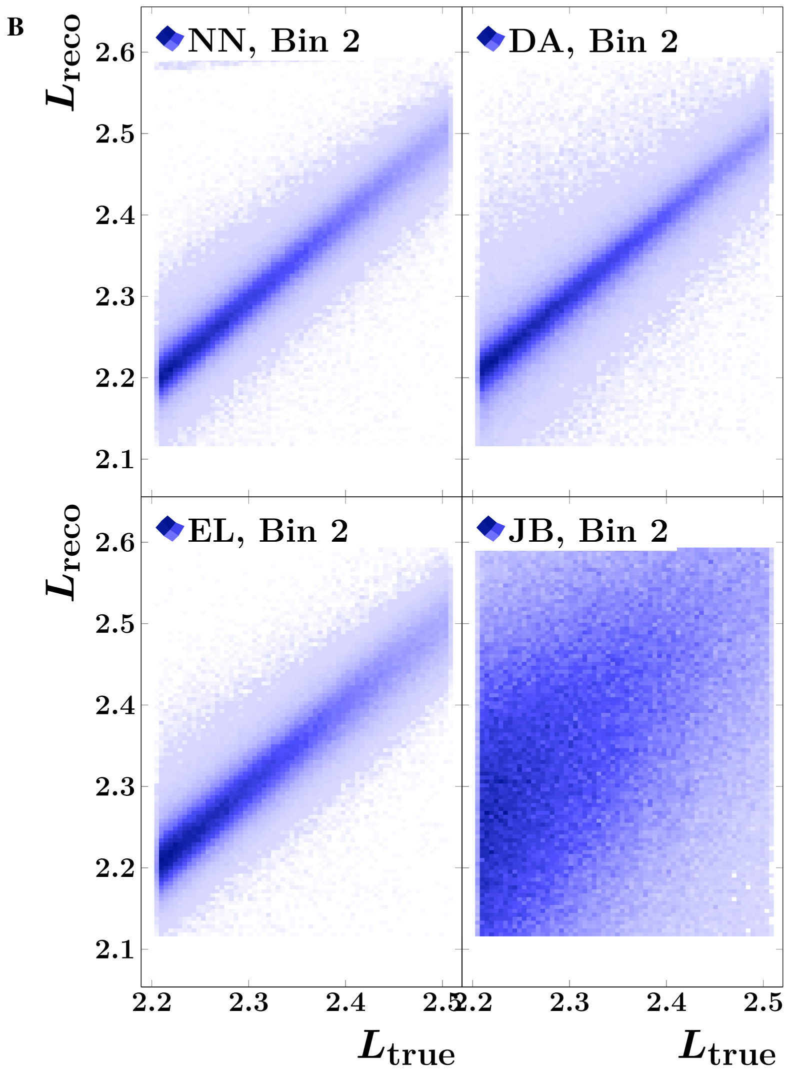


Fig. 39. Continued.

c

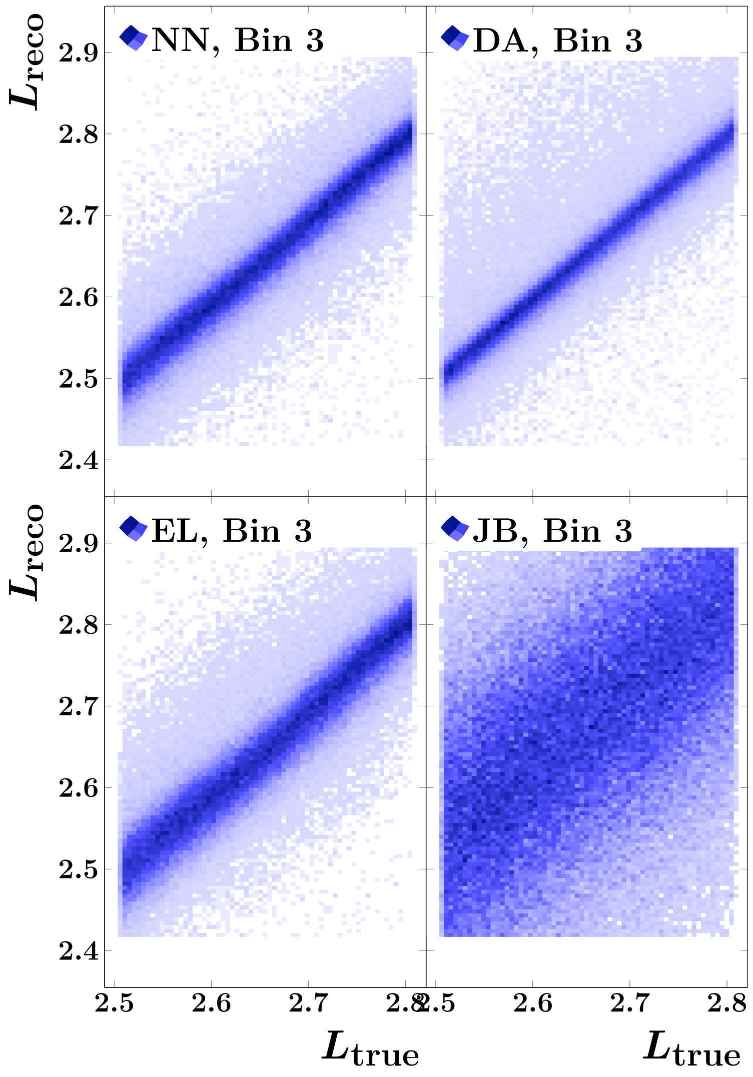


Fig. 39. Continued.

D

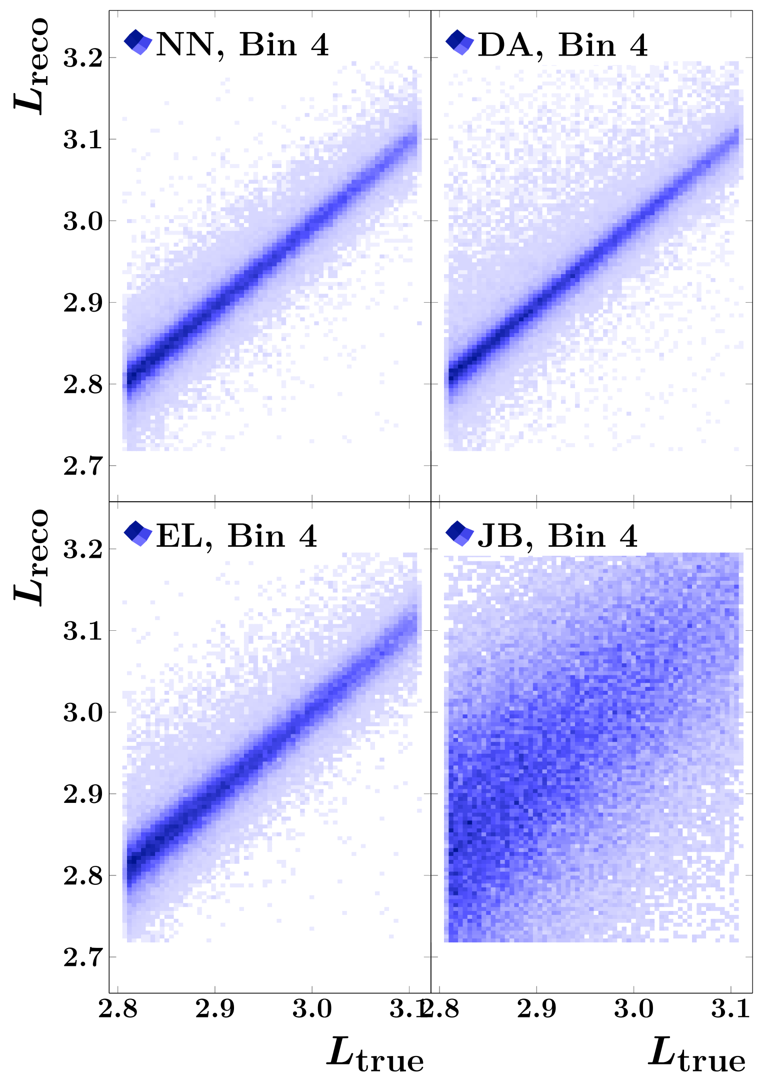


Fig. 39. Continued.

E

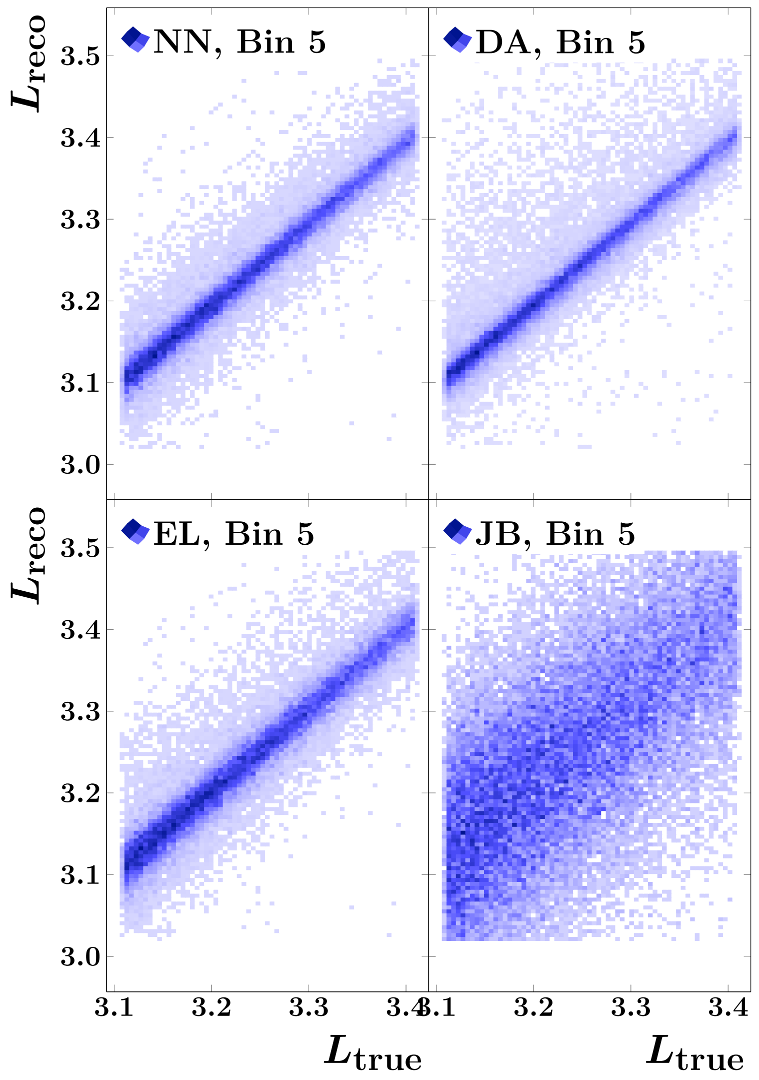


Fig. 39. Continued.

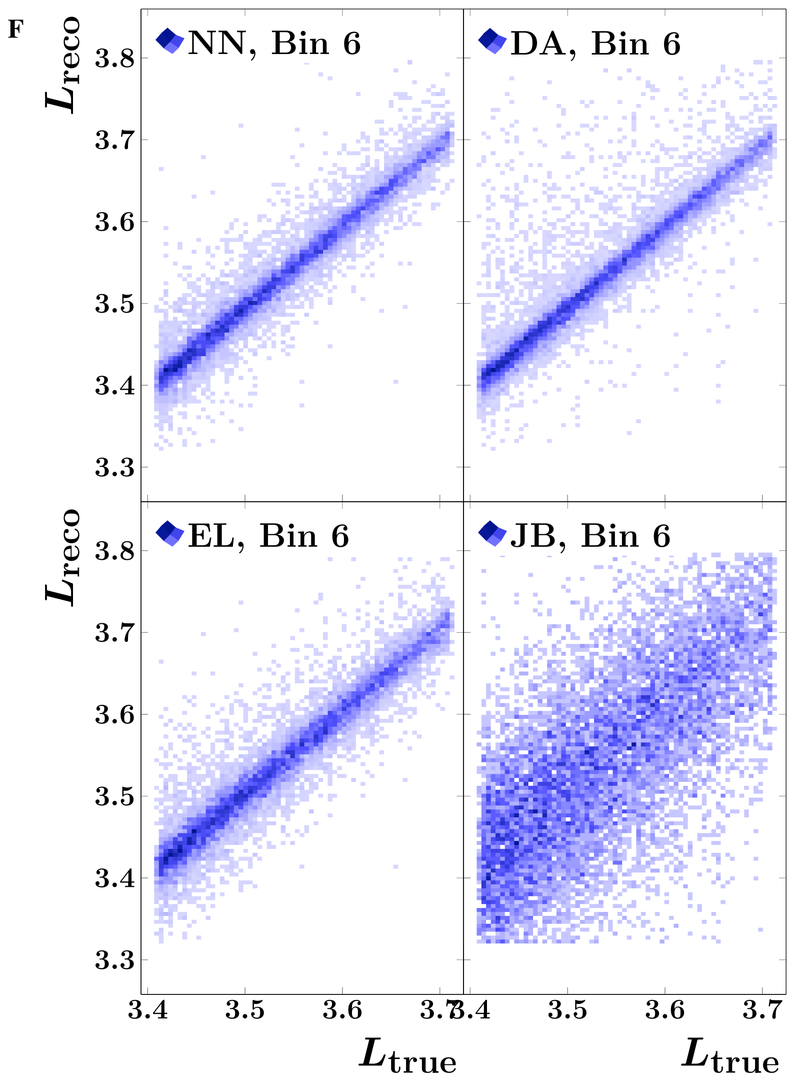


Fig. 39. Continued.

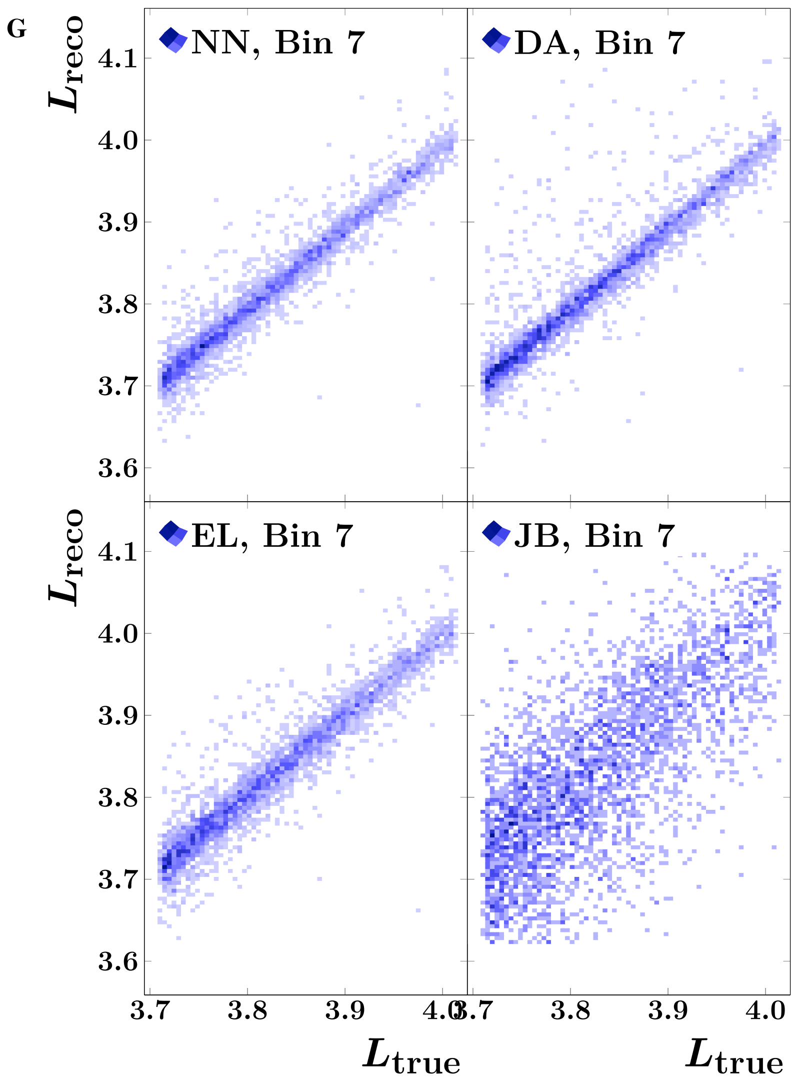


Fig. 39. Continued.

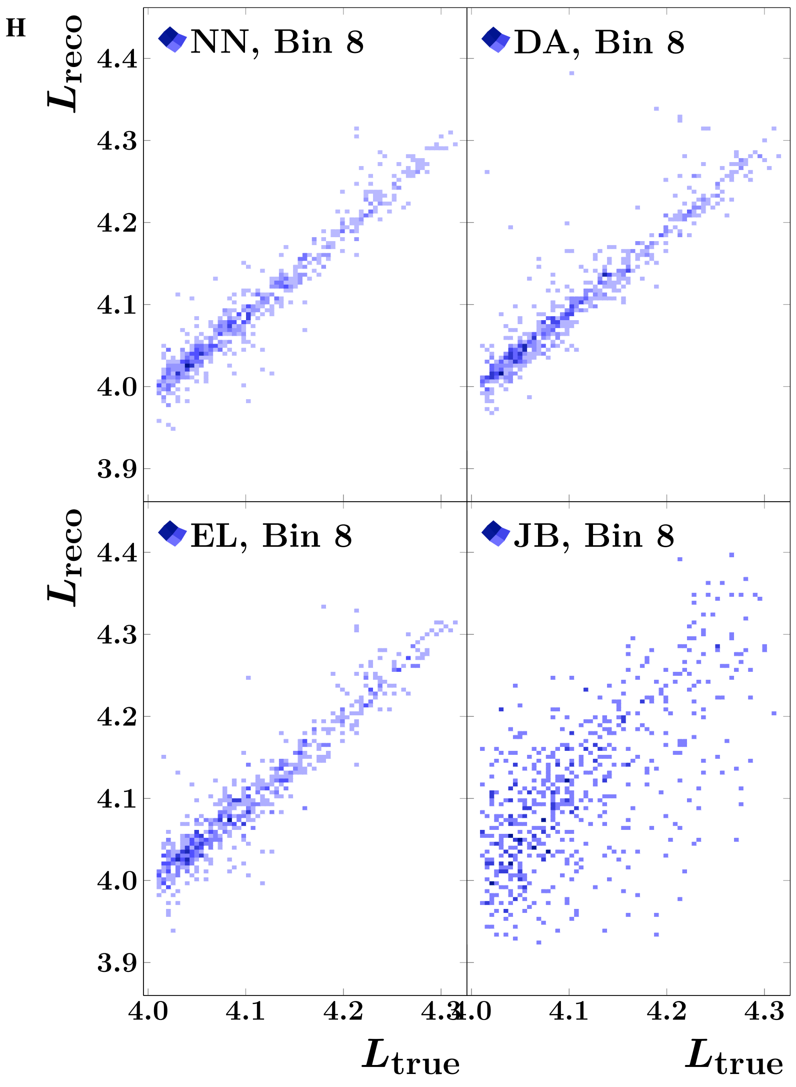


Fig. 39. Continued.

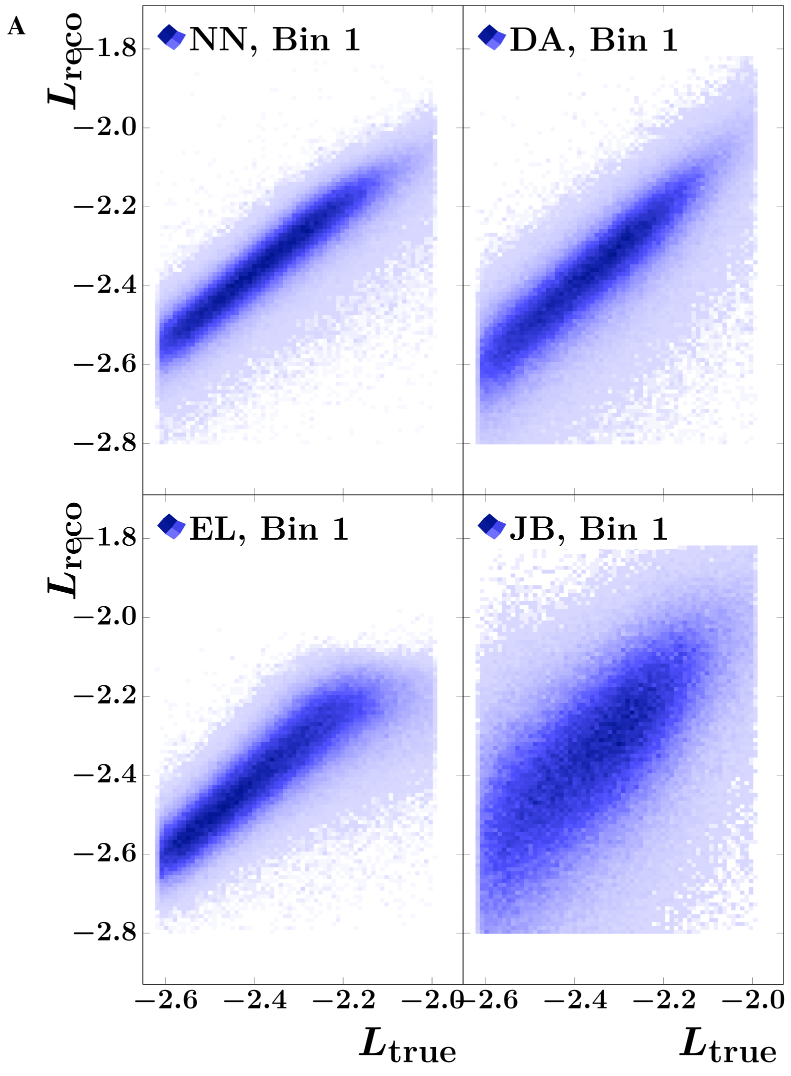


Fig. 40. Distribution of events in $L_{\text{reco}} = \log x$ versus $L_{\text{true}} = \log x_{\text{true}}$ plane for different reconstruction methods in individual analysis bins.

B

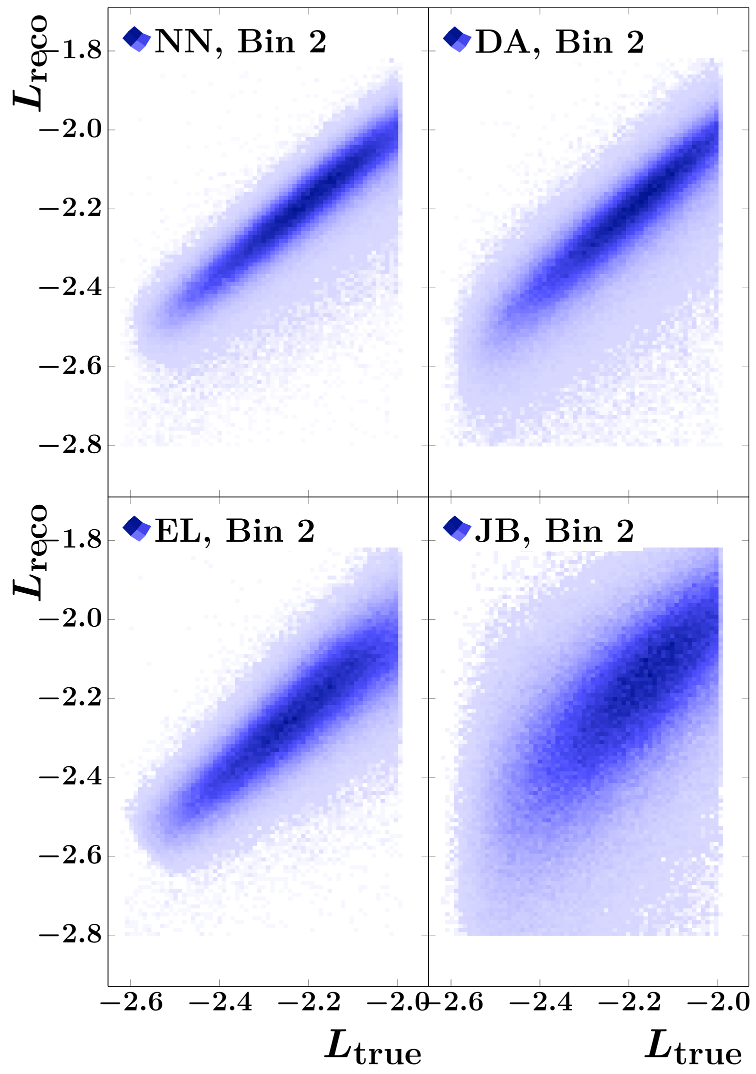


Fig. 40. Continued.

C

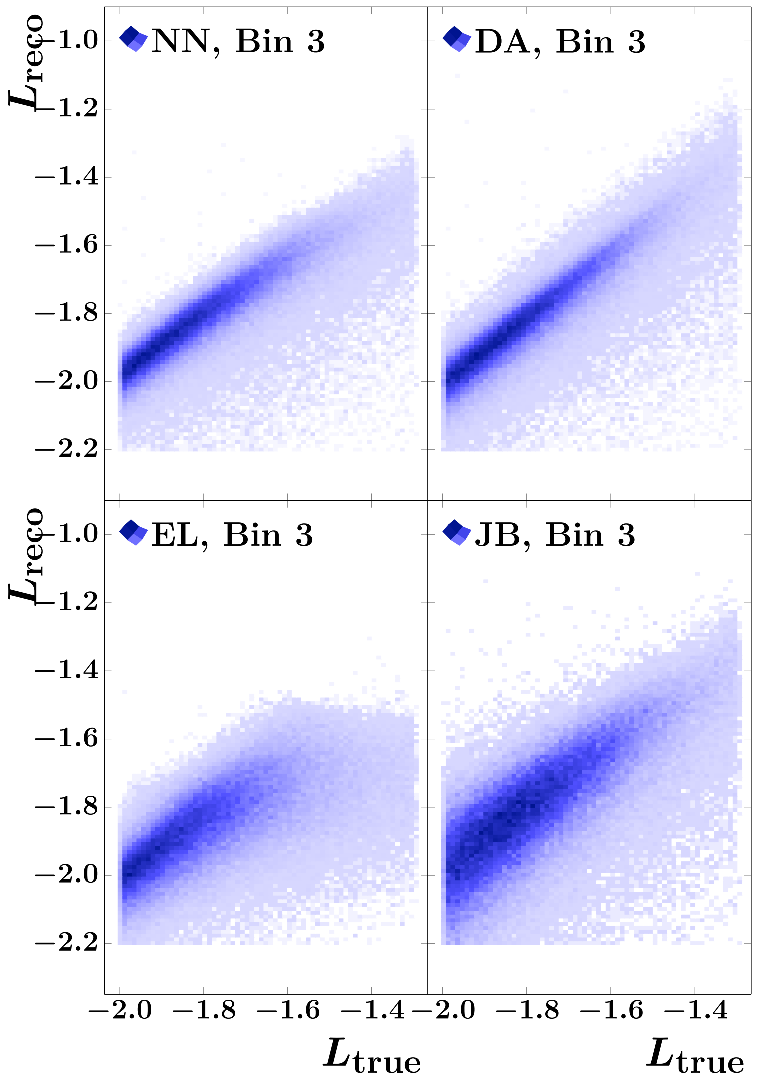


Fig. 40. Continued.

D

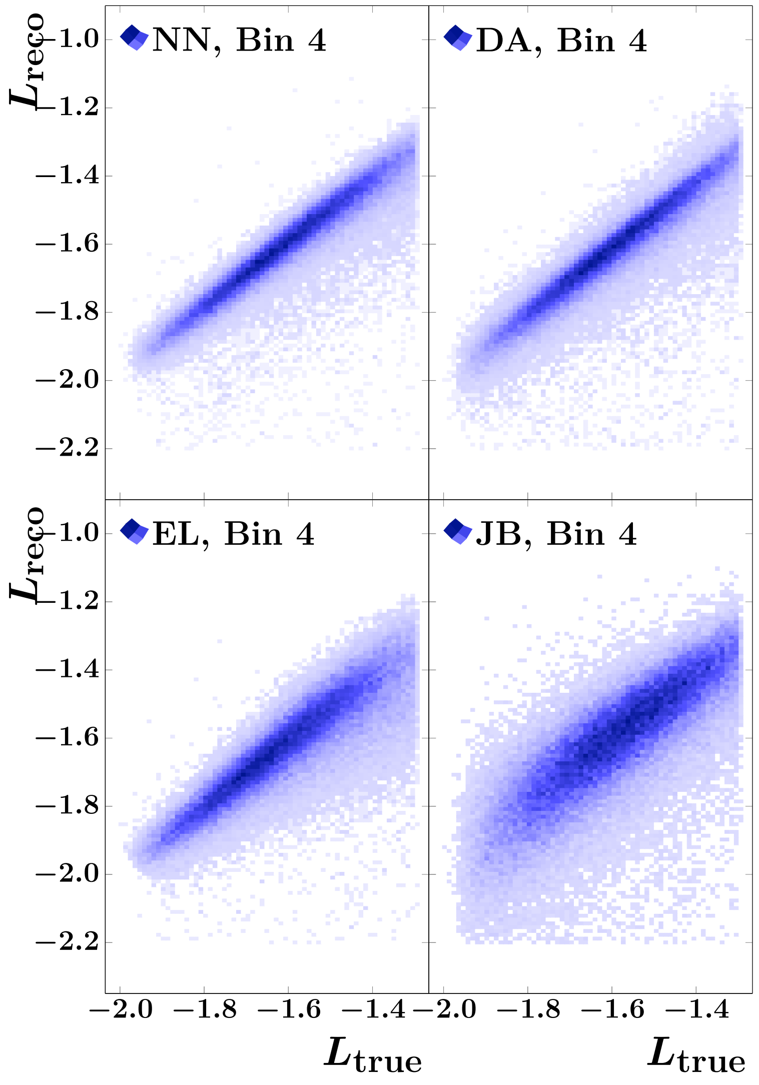


Fig. 40. Continued.

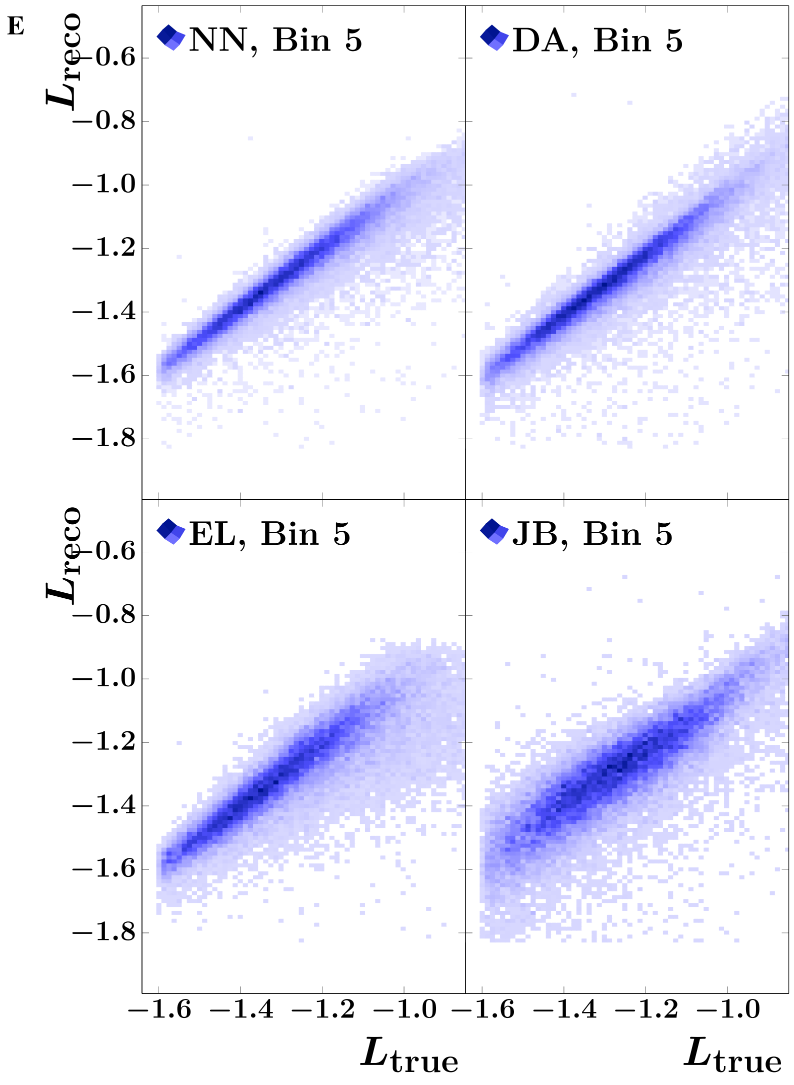


Fig. 40. Continued.

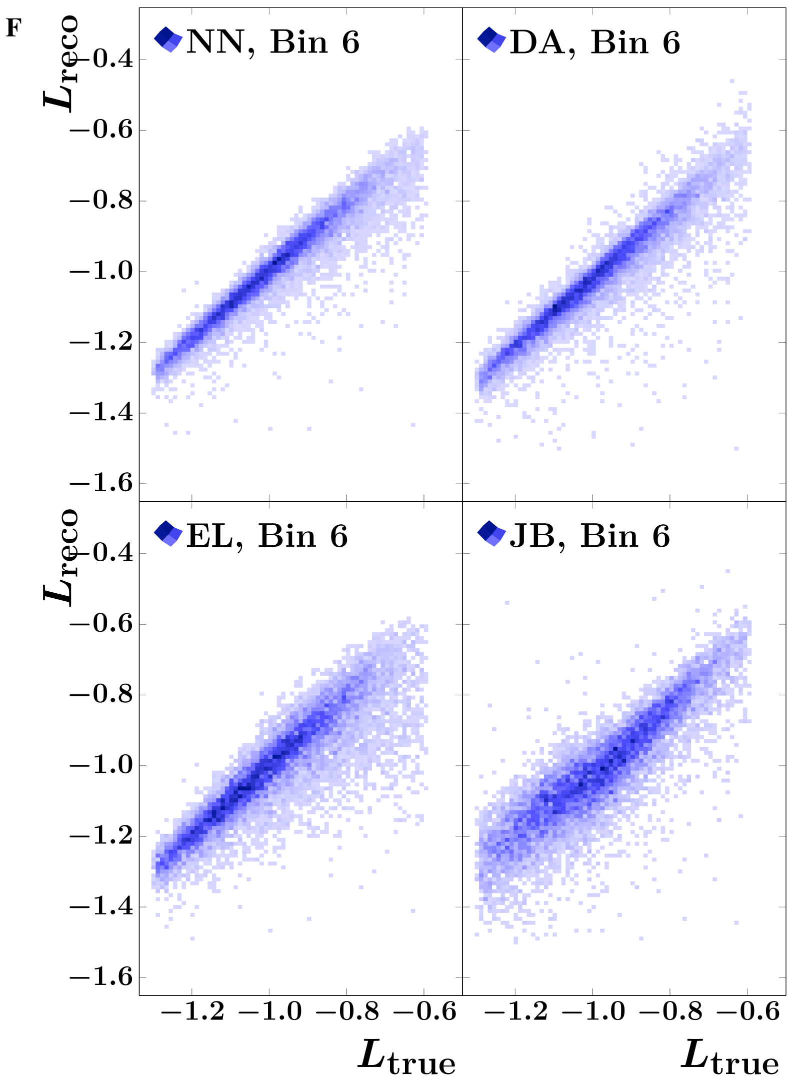


Fig. 40. Continued.

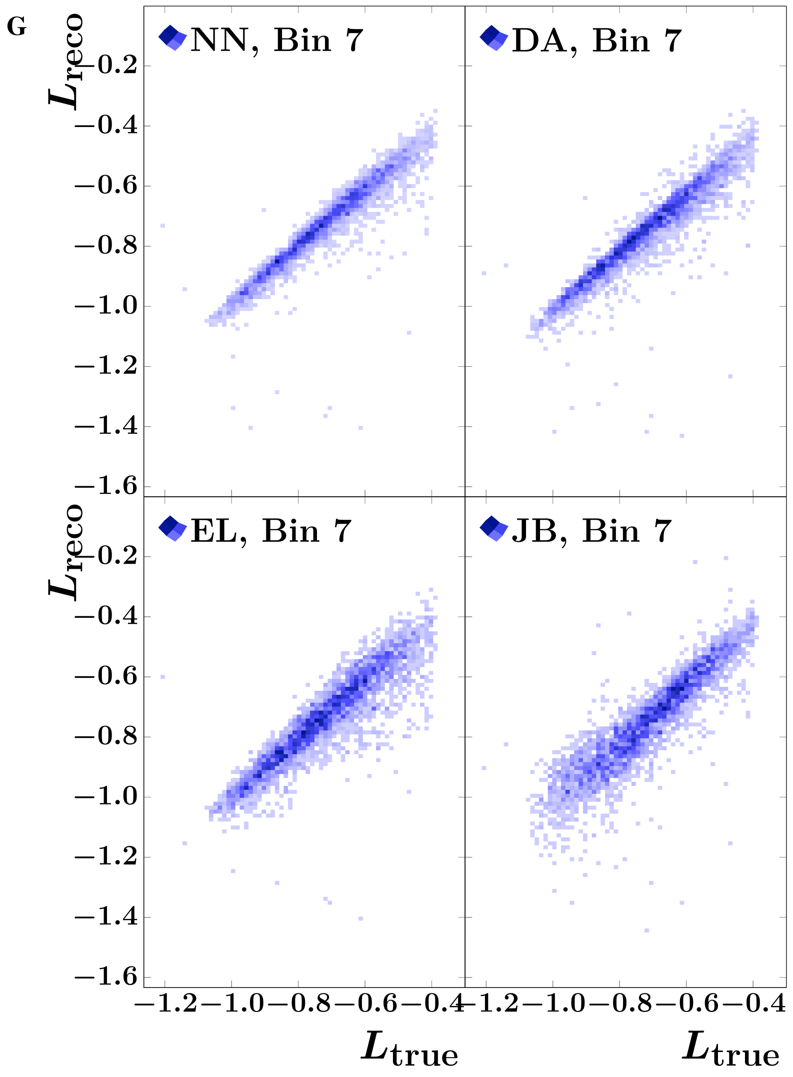


Fig. 40. Continued.

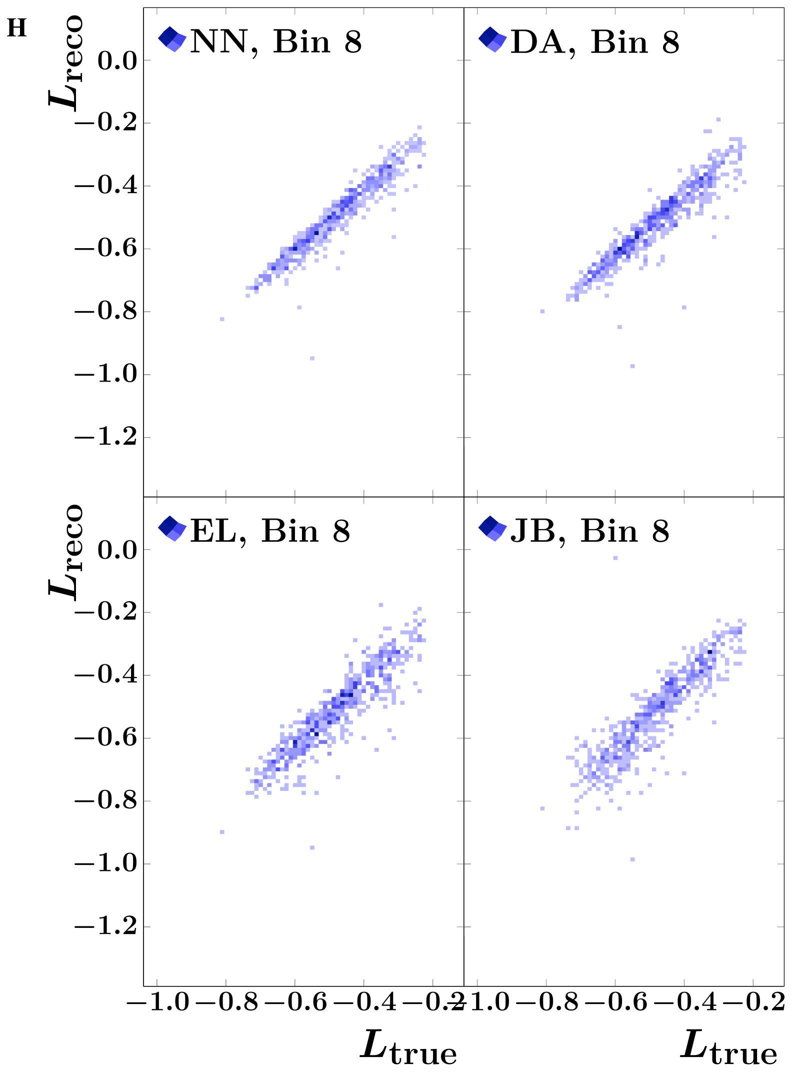


Fig. 40. Continued.

events available elsewhere in the kinematical space and delivers models that perform well even in the bins with highest Q^2 and x.

4.7 CONCLUSIONS

We have presented the use of NN to reconstruct the kinematic observables Q^2 and x in the study of neutral current DIS events at the ZEUS experiment at HERA. The NN models are specially designed to be effective in their universal approximation capability, robust in the sense that increasing the depth of the networks will necessarily reduce empirical error, and computationally efficient with a structure that avoids "vanishing" gradients arising in the backpropagation algorithm.

Compared to the classical reconstruction methods, the NN-based approach enables significant improvements in the resolution of Q^2 and x. At the same time, it is evident that the usage of the NN approach allows to match easily any definition of Q^2 and x at the true level that is preferred for a given physics analysis.

The large samples of simulated data required for the training of the NN can be generated rapidly at modern data centers. Also, NN allow to effectively extract information from large data sets. This suggests that the new approach for the reconstruction of DIS kinematics can serve as a rigorous method to combine and outperform the classical reconstruction methods at ongoing or upcoming DIS experiments. We will extend the approach beyond inclusive DIS measurements and will study next the use of NN for the reconstruction of event kinematics in semi-inclusive and exclusive DIS measurements.

4.8 SOFTWARE USED

The `ROOT` package [92] of version 6.22 was used to read the ZEUS data, provided by the data preservation at Max-Planck for Physics [1], analyze it and prepare plain text (or ROOT files) with selected information to be used with the ML tools. The selected information from the plain text (or ROOT) files was piped using the `pandas` package [103] into `Keras` [104] interface to `tensorflow` [105] 2.3.0 library to train the ML models. The packages `Eigen` [106], `frugally-deep` [107], `JSON for Modern C++` [108] and `FunctionalPlus` [109] were used for execution of the trained models after these were converted into `frugally-deep` model format [107] to be used with C++ application. The dependencies for the `tensorflow` were supplied from the `PyPi` repository. The training was performed using libraries for computing on GPUs from the `CUDA` [110] framework of version 10.1. We are grateful to MPCDF[1] for the ability to compile

[1]Max Planck Computing and Data Facility, Gießenbachstraße 2, 85748 Garching

and execute the codes on the HPC cluster “Cobra” in [111]. The operation system used was Linux on x86_64 architecture using `gcc` [112] of version 7.3 and `python` [113] of version 3.6.8.

The figures with the final results were produced with the `PGFPlots` [114] package.

CHAPTER 5

CONCLUSIONS AND FUTURE WORK

To address computational challenges in learning deep neural networks, properties of deep **ReLU** networks were studied to develop a multi-scale learning model. The multi-scale model was compared to the multi-grade learning models.

Unlike the deep neural network learned from the standard single-scale, single-grade model, the multi-scale neural networks use low scale information from all hidden layers, and thusly provide a robust approximation method that requires fewer parameters, lower computational time, and is resistant to noise. In addition to this, the multi-scale method can overcome the vanishing gradient problem.

It was proven that the collection of multi-scale neural networks are universal approximators in the space of continuous functions.

The neural network learned from a multi-grade model is the superposition of the neural networks, in a stair-shape, each of which is learned from one grade of the learning.

Three proof-of-concept numerical examples presented in the paper demonstrate that the multi-scale and multi-grade methods are superior to the single-scale, single-grade networks.

The extended analysis on reconstructing kinematic observables in deep inelastic scattering kinematics with multi-scale neural networks shows that not only are those models effective for real world problems, but the power of the approximation is sufficiently great, that it can outperform reconstruction methods based on physical laws.

In the future, many more methods of these kinds can be formed to improve the clarity of our inference.

REFERENCES

[1] S. Amerio et. al., “Status Report of the DPHEP Collaboration: A Global Effort for Sustainable Data Preservation in High Energy Physics,” *zenodo*, 2 2015.

[2] I. Goodfellow, Y. Bengio, and A. Courville, “Deep learning,” *MIT Press*, 2016.

[3] Y. LeCun, Y. Bengio, and G. Hinton, “Deep learning,” *Nature*, vol. 521, pp. 436–44, 05 2015.

[4] W. Douglas, “The inside story of how chatgpt was built from the people who made it,” *MIT Technology Review*, 2023.

[5] M. Diefenthaler, A. Farhat, A. Verbytskyi, and Y. Xu, “Deeply learning deep inelastic scattering kinematics,” *Eur. Phys. J. C*, vol. 82, no. 11, p. 1064, 2022.

[6] I. Häggström, C. Schmidtlein, G. Campanella, and T. Fuchs, “Deeppet: A deep encoder-decoder network for directly solving the pet image reconstruction inverse problem,” *Medical Image Analysis*, vol. 54, pp. 253–262, 03 2019.

[7] G. Jiang, J. Wei, Y. Xu, Z. He, H. Zeng, J. Wu, G. Qin, and Y. lu, “Synthesis of mammogram from digital breast tomosynthesis using deep convolutional neural network with gradient guided cgans,” *IEEE Transactions on Medical Imaging*, vol. PP, pp. 1–1, 04 2021.

[8] M. Raissi and G. Karniadakis, “Hidden physics models: Machine learning of nonlinear partial differential equations,” *Journal of Computational Physics*, vol. 357, 08 2017.

[9] G. Wu and H.-I. Suk, “Deep learning in medical image analysis,” *Annual review of biomedical engineering*, vol. 19, 03 2017.

[10] G. Torlai, G. Mazzola, J. Carrasquilla, M. Troyer, R. Melko, and G. Carleo, “Neural-network quantum state tomography,” *Nature Physics*, vol. 14, 05 2018.

[11] Y. Xu and T. Zeng, “Sparse deep neural network for nonlinear partial differential equations,” *Numerical Mathematics: Theory, Methods and Applications*, vol. 16, pp. 58–78, 06 2023.

[12] B. H. R. DeVore and G. Petrova, “Neural network approximation,” *Cambridge University Press*, 2020.

[13] D. Elbrächter, D. Perekrestenko, P. Grohs, and H. Bölcskei, "Deep neural network approximation theory," 2021.

[14] T. Poggio, H. Mhaskar, L. Rosasco, B. Miranda, and Q. Liao, "Why and when can deep-but not shallow-networks avoid the curse of dimensionality: A review," *International Journal of Automation and Computing*, vol. 14, pp. 1–17, 12 2017.

[15] H. Montanelli, H. Yang, and Q. Du, "Deep relu networks overcome the curse of dimensionality for bandlimited functions," 2020.

[16] D. Yarotsky, "Error bounds for approximations with deep relu networks," *Neural Networks*, vol. 94, 10 2016.

[17] H. Montanelli and H. Yang, "Error bounds for deep relu networks using the kolmogorov–arnold superposition theorem," *Neural Networks*, vol. 129, 05 2020.

[18] D.-X. Zhou, "Universality of deep convolutional neural networks," *Applied and Computational Harmonic Analysis*, vol. 48, 06 2019.

[19] I. Daubechies, R. DeVore, S. Foucart, B. Hanin, and G. Petrova, "Nonlinear approximation and (deep) relu networks," 2019.

[20] Y. Wang, "A mathematical introduction to generative adversarial nets (gan)," 2020.

[21] S. Lock, "What is ai chatbot phenomenon chatgpt and could it replace humans?" *The Guardian*, 2022.

[22] L. Bottou and O. Bousquet, "The Tradeoffs of Large-Scale Learning," in *Optimization for Machine Learning*. The MIT Press, 09 2011. [Online]. Available: https://doi.org/10.7551/mitpress/8996.003.0015

[23] L. Bottou, "Online algorithms and stochastic approximations," in *Online Learning and Neural Networks*, D. Saad, Ed. Cambridge, UK: Cambridge University Press, 1998, revised, oct 2012. [Online]. Available: http://leon.bottou.org/papers/bottou-98x

[24] Y. Bengio, P. Lamblin, D. Popovici, H. Larochelle, and U. Montreal, "Greedy layer-wise training of deep networks," in *Advances in Neural Information Processing Systems*, vol. 19, 01 2007.

[25] K. Hornik, M. Stinchcombe and H. White, "Multilayer feedforward networks are universal approximators," *Neural Networks*, vol. 2, no. 5, pp. 359–366, 1989.

[26] Y. Xu and H. Zhang, "Convergence of deep relu networks," *Neurocomputing*, vol. 571, 2021.

[27] R. Girshick, "Fast r-cnn," in *2015 IEEE International Conference on Computer Vision (ICCV)*, 2015, pp. 1440–1448.

[28] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 11 2013.

[29] K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification," *IEEE International Conference on Computer Vision (ICCV 2015)*, vol. 1502, 02 2015.

[30] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 06 2015, pp. 3431–3440.

[31] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, 06 2015.

[32] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei, "Imagenet large scale visual recognition challenge," 2014. [Online]. Available: http://arxiv.org/abs/1409.0575

[33] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv 1409.1556*, 09 2014.

[34] Y. Bengio, P. Simard, and P. Frasconi, "Learning long-term dependencies with gradient descent is difficult," *IEEE transactions on neural networks / a publication of the IEEE Neural Networks Council*, vol. 5, pp. 157–66, 02 1994.

[35] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, ser. Proceedings of Machine Learning Research, Y. W. Teh and M. Titterington, Eds., vol. 9. Chia Laguna Resort, Sardinia, Italy: PMLR, 13–15 May 2010, pp. 249–256.

[36] Y. Lecun, L. Bottou, G. Orr, and K.-R. Müller, "Efficient backprop," in *Lecture Notes in Computer Science*, 08 2000.

[37] A. Saxe, J. Mcclelland, and S. Ganguli, "Exact solutions to the nonlinear dynamics of learning in deep linear neural networks," 2014.

[38] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," *Proceedings of the 32nd International Conference on Machine Learning*, 02 2015.

[39] Y. LeCun, B. E. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. E. Hubbard, and L. D. Jackel, "Backpropagation applied to handwritten zip code recognition," *Neural Computation*, vol. 1, pp. 541–551, 1989. [Online]. Available: https://api.semanticscholar.org/CorpusID:41312633

[40] Y. Xu, "Multi-grade deep learning," 2023.

[41] E. Wong, L. Rice, and Z. Kolter, "Overfitting in adversarially robust deep learning," in *International Conference on Machine Learning*, 2020. [Online]. Available: https://api.semanticscholar.org/CorpusID:211505864

[42] E. Candes, J. Romberg and T. Tao, "Robust Uncertainty Principles : Exact Signal Frequency Information," *IEEE Transactions on Information Theory*, vol. 52, pp. 489–509, 03 2006.

[43] H. Zhang, Y. Xu and J. Zhang, "Reproducing kernel Banach spaces for machine learning," *Journal of Machine Learning Research*, pp. 2741–2775, 12 2009.

[44] D.P. Bertsekas and J. Tsitsiklis, "Gradient convergence in gradient methods with errors," *SIAM Journal on Optimization*, vol. 10, pp. 627–642, 2000.

[45] I. Sutskever et al., "On the importance of initialization and momentum in deep learning," *ICML'13: Proceedings of the 30th International Conference on International Conference on Machine Learning*, vol. 28, p. 1139–1147, 2013.

[46] D. Kingma and J. Ba, "Adam: A method for stochastic optimization," *International Conference on Learning Representations*, 12 2014.

[47] B.H. Wiik et al., "HERA - A Proposal for a Large Electron Proton Colliding Beam Facility at DESY," https://old.inspirehep.net/record/19436/files/Fulltext.pdf, 1981.

[48] B. A. et al., "Volume I. Introduction to DUNE," *JINST*, vol. 15, no. 08, p. T08008, 2020.

[49] F. Gautheron et al., "COMPASS-II Proposal," https://cds.cern.ch/record/1265628/files/SPSC-P-340.pdf, 2010.

[50] J. Arrington *et al.*, “Physics with CEBAF at 12 GeV and Future Opportunities,” 11 2021.

[51] E. C. Aschenauer, R. S. Thorne, and R. Yoshida, “PDG Chapter 18: Structure Functions,” R. L. Workman and Others, Eds. PTEP, 2022.

[52] G. Voss and B. Wiik, “The Electron proton collider HERA,” *Ann. Rev. Nucl. Part. Sci.*, vol. 44, pp. 413–452, 1994.

[53] H. A. et al., “Combination of measurements of inclusive deep inelastic $e^{\pm}p$ scattering cross sections and QCD analysis of HERA data,” *Eur. Phys. J.*, vol. C75, no. 12, p. 580, 2015.

[54] U. Bassler and G. Bernardi, “On the kinematic reconstruction of deep inelastic scattering at HERA: The Sigma method,” *Nucl. Instrum. Meth.*, vol. A361, pp. 197–208, 1995.

[55] R. Abdul Khaleket et al., “Science Requirements and Detector Concepts for the Electron-Ion Collider: EIC Yellow Report,” 3 2021.

[56] U. Holm, “The ZEUS detector: Status report 1993,” http://dx.doi.org/10.3204/PUBDB-2017-12635, p. 597, 1993.

[57] H. Abramowicz, A. Caldwell and R. Sinkus, “Neural network based electron identification in the ZEUS calorimeter,” *Nucl. Instrum. Meth.*, vol. A365, pp. 508–517, 1995.

[58] A. Accardi et al., “Electron Ion Collider: The Next QCD Frontier,” *Eur. Phys. J.*, vol. A52, no. 9, p. 268, 2016.

[59] Junze Liu et al., “Deep-Learning-Based Kinematic Reconstruction for DUNE,” 2020.

[60] I. Abt et al., “The H1 detector at HERA,” *Nucl. Instrum. Meth. A*, vol. 386, pp. 310–347, 1997.

[61] A. Kwiatkowski, H. Spiesberger and H.J. Mohring, “Heracles: An Event Generator for *ep* Interactions at HERA Energies Including Radiative Processes: Version 1.0,” *Comput. Phys. Commun.*, vol. 69, pp. 155–172, 1992.

[62] G. Altarelli and G. Parisi, “Asymptotic Freedom in Parton Language,” *Nucl. Phys.*, vol. B126, pp. 298–318, 1977.

[63] T. Sjöstrand, S. Mrenna and P.Z. Skands, “PYTHIA 6.4 physics and manual,” *JHEP*, vol. 05, p. 026, 2006.

[64] T. Sjöstrand et al., "An introduction to PYTHIA 8.2," *Comput. Phys. Commun.*, vol. 191, pp. 159–177, 2015.

[65] E. Bothmann et al., "Event generation with SHERPA 2.2," *SciPost Phys.*, vol. 7, no. 3, p. 034, 2019.

[66] W. Kilian, T. Ohl and J. Reuter, "WHIZARD: Simulating Multi-Particle Processes at LHC and ILC," *Eur. Phys. J.*, vol. C71, p. 1742, 2011.

[67] J. Bellm et al., "Herwig 7.0/Herwig++ 3.0 release note," *Eur. Phys. J.*, vol. C76, no. 4, p. 196, 2016.

[68] G. Ingelman, A. Edin and J. Rathsman, "LEPTO 6.5: A Monte Carlo generator for deep inelastic lepton - nucleon scattering," *Comput. Phys. Commun.*, vol. 101, pp. 108–134, 1997.

[69] L. Lönnblad, "ARIADNE version 4: a program for simulation of QCD cascades implementing the color dipole model," *Comput. Phys. Commun.*, vol. 71, pp. 15–31, 1992.

[70] H. Jung et al., "The CCFM Monte Carlo generator CASCADE version 2.2.03," *Eur. Phys. J.*, vol. C70, pp. 1237–1249, 2010.

[71] H. Jung, "Hard diffractive scattering in high-energy *ep* collisions and the Monte Carlo generator RAPGAP," *Comput. Phys. Commun.*, vol. 86, pp. 147–161, 1995.

[72] F. Cascioli, P. Maierhofer and S. Pozzorini, "Scattering Amplitudes with OpenLoops," *Phys. Rev. Lett.*, vol. 108, p. 111601, 2012.

[73] Z. Bern et al., "The BlackHat library for one-loop amplitudes," *J. Phys. Conf. Ser.*, vol. 523, p. 012051, 2014.

[74] J. Alwall et al., "MadGraph 5 : Going Beyond," *JHEP*, vol. 06, p. 128, 2011.

[75] R. Brun et al., "GEANT3," https://cds.cern.ch/record/1119728/files/CERN-DD-EE-84-1.pdf, 1987.

[76] S. A. et al., "GEANT4–a simulation toolkit," *Nucl. Instrum. Meth.*, vol. A506, pp. 250–303, 2003.

[77] S. Bentvelsen, J. Engelen and P. Kooijman, "Reconstruction of (x,Q^2) and extraction of structure functions in neutral current scattering at HERA," in *Workshop on Physics at HERA Hamburg, Germany, October 29-30, 1991*, 1992, pp. 23–42.

[78] J. Currie et al, “NNLO QCD corrections to jet production in deep inelastic scattering,” *JHEP*, vol. 07, p. 018, 2017, [Erratum: JHEP 12, 042 (2020)].

[79] V. Andreev *et al.*, “Measurement of Lepton-Jet Correlation in Deep-Inelastic Scattering with the H1 Detector Using Machine Learning for Unfolding,” *Phys. Rev. Lett.*, vol. 128, no. 13, p. 132002, 2022.

[80] A. Arbuzov el al., “Hector 1.00: A Program for the calculation of QED, QCD and electroweak corrections to ep and $lepton^{\pm}N$ deep inelastic neutral and charged current scattering,” *Comput. Phys. Commun.*, vol. 94, pp. 128–184, 1996.

[81] T. Liu et al., “A new approach to semi-inclusive deep-inelastic scattering with QED and QCD factorization,” *JHEP*, vol. 11, p. 157, 2021.

[82] H. Abramowicz et al., “Inclusive-jet cross sections in NC DIS at HERA and a comparison of the kT, anti-kT and SIScone jet algorithms,” *Phys. Lett. B*, vol. 691, pp. 127–137, 2010.

[83] J. Malka and K. Wichmann, “The ZEUS data preservation project,” *J. Phys. Conf. Ser.*, vol. 396, p. 022033, 2012.

[84] A. Verbytskyi, “The ZEUS long term data preservation project,” *PoS*, vol. DIS2016, p. 264, 2016.

[85] K. He et al., “Deep Residual Learning for Image Recognition,” *Computer Vision and Pattern Recognition*, 2015.

[86] Y. Xu and H. Zhang, “Convergence of deep convolutional neural networks,” *Neural Networks*, vol. 153, pp. 553–563, 2022.

[87] A. Krizhevsky et al., “ImageNet Classification with Deep Convolutional Neural Networks,” in *Advances in Neural Information Processing Systems 25*, F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, Eds. Curran Associates, Inc., 2012, pp. 1097–1105. [Online]. Available: http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf

[88] G.A. Schuler and H. Spiesberger, “DJANGO: The Interface for the event generators HERACLES and LEPTO,” in *Workshop on Physics at HERA*, 1991.

[89] H. L. et al., “Global QCD analysis of parton structure of the nucleon: CTEQ5 parton distributions,” *Eur. Phys. J.*, vol. C12, pp. 375–392, 2000.

[90] M. Bengtsson and T. Sjostrand, "Parton Showers in Leptoproduction Events," *Z. Phys.*, vol. C37, p. 465, 1988.

[91] G. Gustafson and U. Pettersson, "Dipole Formulation of QCD Cascades," *Nucl. Phys.*, vol. B306, pp. 746–758, 1988.

[92] I. Antcheva et al., "ROOT: A C++ framework for petabyte data storage, statistical analysis and visualization," *Comput. Phys. Commun.*, vol. 182, pp. 1384–1385, 2011.

[93] J. B. et al., "Measurement of high Q^2 neutral current e^+p deep inelastic scattering cross-sections at HERA," *Eur. Phys. J.*, vol. C11, pp. 427–445, 1999.

[94] S. C. et al., "Jet-radius dependence of inclusive-jet cross-sections in deep inelastic scattering at HERA," *Phys. Lett.*, vol. B649, pp. 12–24, 2007.

[95] S. Chekanov et al., "Event shapes in deep inelastic scattering at HERA," *Nucl. Phys.*, vol. B767, pp. 1–28, 2007.

[96] S. C. et al., "Inclusive-jet and dijet cross-sections in deep inelastic scattering at HERA," *Nucl. Phys.*, vol. B765, pp. 1–30, 2007.

[97] H. A. et al., "Inclusive-jet cross sections in NC DIS at HERA and a comparison of the kT, anti-kT and SIScone jet algorithms," *Phys. Lett.*, vol. B691, pp. 127–137, 2010.

[98] ——, "Inclusive dijet cross sections in neutral current deep inelastic scattering at HERA," *Eur. Phys. J.*, vol. C70, pp. 965–982, 2010.

[99] D. Lontkovskyi, "Measurement of Jet Production with the ZEUS Detector," Ph.D. dissertation, Hamburg U., 2015, http://ediss.sub.uni-hamburg.de/volltexte/2016/7801/.

[100] H. Perrey, "Jets at low Q^2 at HERA and radiation damage studies for silicon sensors for the XFEL," Ph.D. dissertation, Hamburg U., 2011.

[101] S. Smith, P. Kindermans and Q. Le, "Don't Decay the Learning Rate, Increase the Batch Size," 2017.

[102] R. Aggarwal, "Measurement of high x neutral current ep cross sections and extractions of xF^3 structure function using ZEUS detector at HERA," Ph.D. dissertation, Panjab University, 2012, http://hdl.handle.net/10603/80282.

[103] W. McKinney, "Data Structures for Statistical Computing in Python," in *Proceedings of the 9th Python in Science Conference*, Stéfan van der Walt and Jarrod Millman, Eds., 2010, pp. 56 – 61.

[104] F. Chollet et al., "Keras," https://keras.io, 2015.

[105] M. Abadi et al., "TensorFlow: Large-scale machine learning on heterogeneous systems," http://tensorflow.org/, 2015.

[106] G. Guennebaud, B. Jacob et al., "Eigen v3," http://eigen.tuxfamily.org, 2021.

[107] T. Hermann, "frugally-deep," https://github.com/Dobiasd/frugally-deep, 2021.

[108] N. Lohmann, "JSON for Modern C++ ," https://github.com/nlohmann/json, 2021.

[109] T. Hermann, "FunctionalPlus," https://github.com/Dobiasd/FunctionalPlus, 2021.

[110] NVIDIA Corp., P. Vingelmann and F.H.P Fitzek, "Cuda, release: 10.2.89," https://developer.nvidia.com/cuda-toolkit, 2020.

[111] Max-Planck Computing and Data Facility, "About the HPC system Cobra," https://www.mpcdf.mpg.de/services/supercomputing/cobra, 2021.

[112] R.M. Stallman and GCC Developer Community, *Using The Gnu Compiler Collection: A Gnu Manual For GCC Version 4.3.3*. Paramount, CA: CreateSpace, 2009.

[113] Python Software Foundation, "Python language reference, version 3," http://www.python.org, 2019.

[114] C. Feuersänger, "Pgfplots," http://pgfplots.sourceforge.net/, 2021.

VITA

Abdullah Ayar Farhat
Department of Computational and Applied Mathematics
Old Dominion University
Norfolk, VA 23529

EDUCATION

2009-2013, B.A. Mathematics, Johns Hopkins University
2012-2013, M.A. Mathematics, Johns Hopkins University

PROFESSIONAL EXPERIENCE

2013-2025, Teacher, New Horizon's Governor's School for Science and Technology
2015, Adjunct Professor, Hampton University
2015-2018, Adjunct Professor, Thomas Nelson Community College

CONFERENCE PRESENTATIONS

1. Streaming Readout Workshop VII, INDRA-ASTRA - Prototype for automated data-quality monitoring and autocalibrations, Nov. 2020
2. XXIX International Workshop on Deep-Inelastic Scattering and Related Subjects, Deeply learning deep inelastic scattering kinematics, May 2022

PUBLICATIONS

1. M. Diefenthaler, A. Farhat, A. Verbytskyi, and Y. Xu, "Deeply learning deep inelastic scattering kinematics," Eur. Phys. J. C, vol. 82, no. 11, p. 1064, 2022.