Available online at www.sciencedirect.com

ScienceDirect

Procedia Manufacturing 53 (2021) 680–689

Procedia MANUFACTURING

www.elsevier.com/locate/procedia

49th SME North American Manufacturing Research Conference, NAMRC 49, Ohio, USA

Estimating Johnson–Cook Material Parameters using Neural Networks

Nesar Ahmed Titu[a], Matt Baucum[a], Timothy No[b], Mitchell Trotsky[b], Jaydeep Karandikar[c], Tony L Schmitz[b,c], Anahita Khojandi[a]

[a]*Department of Industrial and Systems Engineering, University of Tennessee, Knoxville, TN 37996, USA*
[b]*Department of Mechanical, Aerospace, and Biomedical Engineering, University of Tennessee, Knoxville, TN 37996, USA*
[c]*Manufacturing Science Division, Oak Ridge National Laboratory, Oak Ridge, TN, 37831 USA*

Abstract

The five-parameter Johnson-Cook (J-C) material model represents the behavior of a material under extreme mechanical loading, including high temperatures, strains, and strain rates. The goal of this study is to estimate five J-C material parameters and chip thickness jointly for a given set of force components, power, and temperature. The approach uses two neural network models on a dataset simulated by finite element analysis for orthogonal cutting of aluminum 6061-T6. The first model develops a function approximator to predict the force components, power, and temperature using a given set of J-C parameters and chip thickness for aluminum 6061-T6. The second model searches the input space of the first model to estimate the J-C parameter values and chip thickness, given a set of targeted force components, power, and temperature of interest. The performance of both neural network models is evaluated using mean absolute percentage error. The results suggest that the developed neural networks-based approach is capable of estimating multiple J-C parameters and chip thickness that will result in a targeted force components, power, and temperatures of interest, given starting 'educated guesses' about these values.

Keywords: Milling force; finite element analysis; Johnson-Cook; Machine learning; Neural Networks

1. Introduction

The Johnson-Cook (J-C) model consists of five parameters that are used to describe the behavior of the material at high strain (forces) and high temperature [1]. Estimating J-C parameters for the given temperature and forces allow us to improve machining productivity by enabling finite element simulation of the cutting process [2].

In the past, researchers have taken different approaches to estimate J-C parameters, including experimental and non-experimental approaches. Experimental approaches use quasi-static tests and dynamic tests at different temperatures and forces to identify J-C constants [3]. For instance, Khan et al. [4] used similar quasi-static and dynamic loading on different temperatures and strain rates along with predictions by modified Khan-Huang-Liang (KHL) viscoplastic constitutive model to identify the J-C constants. The Split-Hopkinson pressure bar (SHPB) test [5] is a common experimental approach for identifying J-C model parameters.

In contrast to experimental approaches, non-experimental approaches estimate J-C parameters using purely numerical and/or analytical approaches. For instance, Ning and Liang [6] proposed an analytical method for estimating J-C parameters based on the chip formation model in the orthogonal cutting and J-C model where forces and temperatures were used as inputs. The authors used an algorithm based on the mathematical equation of the chip formation model and J-C flow stress model. Later, estimated parameter values were validated by comparing their reference values from SHPB tests in literature. The same

* Corresponding author. Tel.: +1-865-974-0234
E-mail address: khojandi@utk.edu (Anahita Khojandi).

This manuscript has been authored in part by UT-Battelle, LLC, under contract DE-AC05-00OR22725 with the US Department of Energy (DOE). The US government retains and the publisher, by accepting the article for publication, acknowledges that the US government retains a nonexclusive, paid-up, irrevocable, worldwide license to publish or reproduce the published form of this manuscript, or allow others to do so, for US government purposes. DOE will provide public access to these results of federally sponsored research in accordance with the DOE Public Access Plan (http://energy.gov/downloads/doe-public-access-plan). The authors acknowledge support from the Oak Ridge Institute at the University of Tennessee (ORI@UT) Seed funding program and Science Alliance, The University of Tennessee.

10.1016/j.promfg.2021.06.082

authors [7] developed another analytical model based on the chip formation model and an iterative gradient search method to identify J-C parameters using temperature and material properties. Furthermore, to validate the derived J-C parameters, they predicted machining forces using derived J-C parameters and then compared those with the respective forces obtained from the experiment. Klocke et al. [8] used a similar iterative gradient search method with temperature and force measurements to identify and validate the parameters for orthogonal cutting of AISI 1045 and Inconel 718.

Due to the fact that experimental approaches are time-consuming and costly, and per the availability of data by simulation, researchers have recently begun to use numerical approaches combined with experiment-based or simulated data [9]. The numerical approach generally acquires data by Finite Element Analysis (FEA) [10]. The accuracy of the numerical solution depends on the methods and the equations used to determine the J-C parameters [11]. However, numerical approaches may show poor prediction results [12, 13].

Various algorithms have been developed to leverage FEA-simulated data to obtain J-C parameters. For instance, Shrot and Baker applied the Levenberg–Marquardt (L-M) search algorithm [14] to identify J-C parameters based on FEA-simulated data by minimizing the squared error between the FEA-simulated and L-M algorithm-generated J-C parameters. This method successfully estimated the J-C parameters and provided small errors between the parameter values and ground truth values. However, only three of the five parameters were calculated as identifying all five parameters is difficult. This is because different parameter sets can yield similar changes to cutting forces, implying a many-to-one relationship [15]. Furthermore, the L-M search algorithm is a single-objective method that estimates each of the three parameters separately, hence ignoring the possible interaction between the parameters themselves. In another, more comprehensive approach, a weighted multi-objective optimization method [16] was implemented that used sets of data from quasi-static and split Hopkinson pressure bar (quasi-static and SHPB compression) experiments with varying temperature and strain rates to identify all five parameters by the same optimization model. Although this proposed method is generally faster than other analytical methods, it still requires multiple SHPB experiments to determine the parameters, rendering the approach experimentally expensive and time-consuming.

In another study, Majzoobi et al. [17] combined experimental, numerical, and analytical methods to estimate the J-C parameters. They conducted experiments using SHPB and simulation by FEA. They obtained the parameters by the analytical optimization method [18]. The objective function minimized the difference between numerical predictions and experiments. Although the results are promising, this study also used SHPB tests, which are time consuming and computationally expensive. Hence, overall, although several types of research have been conducted, the task of estimating J-C parameters remains challenging due to high experimental cost and mathematical and computational complexity.

Recently, data-driven machine learning (ML) methods have gained popularity and shown great promise in machining processes analysis [19]. This is primarily because ML methods are capable of managing high-dimensional, multi-variate data, and have the capacity of obtaining information from a complex, even disorganized environment [20, 21]. Specifically, Neural Networks (NN), have been successfully used as function approximators to bypass the need for future simulations, given that the NNs are trained on historical simulation data. For instance, Bobbili et al. [22] estimated flow stress of 7017 aluminium alloy under high strain using a NN model. Similarly, many recent works use NN model to predict the deformation behavior of different materials, bypassing computationally expensive alternatives [22, 23, 24].

One of the main challenges in estimating J-C parameters is due to the many-to-one relationship. That is, although given a set of J-C parameters, the resulting temperature and force can be accurately estimated using function approximators, such as NN, the reverse is not necessarily the case. Hence, it is much more difficult to estimate J-C parameters accurately based on a given target temperature and force in practice. One method to solve this problem is response surface methodology (RSM). In ML, RSM [25] works in the case of many-to-one relationship by finding patterns and relations between several inputs and one or more outputs. Many studies of machining processes use RSM to estimate machining parameters [26, 27]. For example, Malakizadi et al. [28] used the RSM method to identify the flow stress in metal cutting process and estimate J-C parameters. They obtained better prediction results than many of the previous works. However, computational cost of iterative FEA simulations causes this approach to remain inefficient in practice.

In this study, we develop a NN-based framework to estimate J-C parameters. Different from past work (e.g., [14]) we develop a multivariate regression model to simultaneously estimate all J-C parameters, hence accounting for the interactions between the parameters. In addition, different from past works that use RSM to address the many-to-one relationship, in this study we devise an approach that relies on the timing of training in different layers of a NN. Our proposed approach allows for an efficient, reliable, and accurate method to estimate the J-C parameters from given target temperature and force.

The rest of the manuscript is organized as follows. Section 2 discusses the data used in this study and describes the methods and models developed in detail. Section 3 presents the results and describes the validation. Section 4 provides a discussion and expands on the future works. Finally, Section 5 concludes the study.

2. Methods

In this study, we develop a method to estimate J-C parameters. Specifically, we develop two NN models, including a *forward model* and a *backward model*. In the forward model, a multivariate NN regression model predicts the four-tuple output variables (including force components, power, and temperature) from six-tuple input variables (five J-C parameters and

chip thickness). Consequently, the backward model, which is also a NN, specifically a modified version of forward model, estimates five-tuple J-C parameters and chip thickness using four-tuple outputs.

2.1. Data

The Johnson-Cook flow stress model has been widely studied in the literature. See Eq. 1, where σ is the flow stress, ε is the equivalent plastic strain, $\dot{\varepsilon}$ is the strain rate, T is the deformation temperature, and T_m is the melt temperature. The material constants are A, B, n, C and m, where A is the yield strength of the material under reference conditions, B is the strain hardening constant, n is the strain hardening coefficient, C is the strain rate strengthening coefficient, and m is the thermal softening coefficient. Also, $\dot{\varepsilon_{\mathbf{ref}}}$ and T_{ref} are the reference strain rate and the reference deformation temperature.

$$\sigma = (A + B\varepsilon^n)(1 + C\ln(\ln(\tfrac{\dot{\varepsilon}}{\dot{\varepsilon}_{ref}}))(1 - (\tfrac{T - T_{ref}}{T_m - T_{ref}}^m)) \quad (1)$$

Multiple authors report the Eq. 1 material constants for 6061-T6 aluminum [29, 30, 31, 32, 33, 34, 35, 36, 37, 38], the workpiece material selected for this study. A summary of these values is provided in Table 1.

Table 1: Johnson-Cook flow stress model parameters. The last two rows show the mean and standard deviation (SD), respectively.

Reference	A (MPa)	B (MPa)	C	n	m
29	324	114	0.0020	0.4200	1.3400
29	250	79.7	0.0249	0.4990	1.499
30	293.4	121.2	0.0020	0.2300	1.3400
31	324.1	113.8	0.0020	0.4200	1.3400
32	250	70	0.0010	0.4990	1
32	250	70	0.0010	0.4990	1.3150
32	250	79	0.0249	0.4990	1.4990
32	250	137	0.0205	0.4990	1.4990
32	250	209	0.0010	0.4990	1.4990
33	275	86		0.3900	1
33	324	114	0.0020	0.4200	1.3400
33	335	85	0.0120	0.1100	1
33	250	79.7	0.0249	0.4990	1.4990
34	324	114	0.0020	0.4200	1.3400
35	236.7	41.2	0.0411	0.0840	1.4100
35	293.4	121.2	0.0020	0.2300	1.3400
35	324	114	0.0020	0.4200	1.3400
36	275	86	0.0031	0.3900	1
36	324	114	0.0020	0.4200	1.3400
37	324	114	0.0020	0.4200	1.3400
37	164	211	0.0019	0.4650	1.4190
38	293	121.2	0.0020	0.2300	1.3400
38	324	114	0.0020	0.4200	1.3400
Mean	282.9	109.1	0.0081	0.3905	1.3210
SD	42.9	39.2	0.0114	0.1252	0.1640

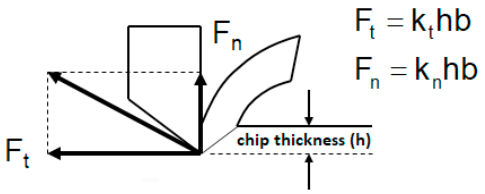


Fig. 1: Orthogonal cutting force model.

The final two rows of Table 1 give the mean and standard deviation values for the five J-C parameters. Values for A, B, C, n, and m are randomly sampled from normal distributions centered at the mean value with one standard deviation (Table 1). We use the data generated from an FEA software called AdvantEdge. FEA is a computer-simulated numerical method for predicting the behavior of physical phenomena, such as machining. AdvantEdge is a popular FEA software used to model machining processes and predict cutting performance. It provides an alternative to trial-and-error cutting tests.

The sampled 6061-T6 aluminum Johnson-Cook material model is defined manually in AdvantEdge. The software runs the orthogonal cutting simulation to the specified cutting parameters. The simulation parameters that are varied are the five J-C coefficients (A, B, C, n, m), chip thickness (h). The following orthogonal cutting parameters are specified: cutting speed (480 m/min); chip width (1 mm); and variable chip thickness (0.13, 0.14, 0.15, 0.16, 0.17 mm). Each set of J-C parameters is used for a simulation at each of the five chip thickness values. A linear regression is performed to extract the slope, which represents the cutting force coefficient that relates the force component to the chip area (i.e., the product of chip thickness and width). The friction coefficient is set to a fixed value of 0.5. The outputs are tangential force (F_t), normal force (F_n), power (P), and temperature (T). The outputs are measured across their steady-state portion to provide an average value. The cutting force components (F_t, F_n) are the products of the cutting force coefficient (k_t, k_n), the chip thickness (h), and the chip width (b).The chip thickness is shown in the Fig. 1. The chip width dimension is into the page/computer screen.

Overall, the dataset consists of 1,000 observations (i.e., 200 sets of sampled J-C parameters for the five chip thicknesses) with ten variables (see the appendix). The input variables are five J-C parameters and chip thickness (h), and the output variables include F_t, F_n, P, and T.

2.2. Neural Networks: Forward and Backward Approaches

We provide a brief theoretical overview of our neural network approach for J-C parameter estimation. An artificial neural network or simply NN is an algorithmic series that finds underlying patterns in a set of data. It is comprised of an input layer, a series of hidden layers, and an output layer. Given data, NNs learn to estimate/predict the output data using the input data. This is done through adjusting a series of weights and bias values in each layer of the network and applying ac-

tivation functions. Specifically, each layer of a NN consists of many neurons. Every neuron in a given layer is connected to all neurons in the next layer with a weight. The value of each neuron in a layer is the summation of all neurons from the previous layer multiplied by their corresponding weights, plus a bias, to which an activation function is applied.

Fig. 2 provides a simple NN model with an input layer consisting of three neurons, a hidden layer consisting of two neurons, and an output layer of one neuron. The weights $W_1, \ldots, W_8$ indicate the weights of each connection to the next neuron. The value $X_{2,1}$ is calculated as $X_{2,1} = f(X_{1,1} \cdot W_1 + X_{1,2} \cdot W_3 + X_{1,3} \cdot W_5 + b)$ where $f(\cdot)$ is the activation function and b is bias. Similarly, $X_{2,2} = f(X_{1,1} \cdot W_2 + X_{1,2} \cdot W_4 + X_{1,3} \cdot W_6 + b)$.

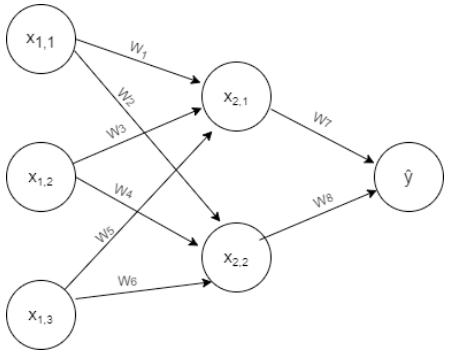


Fig. 2: A simple NN model with three layers.

For our purposes, we refer to any such neural network that is trained to predict a set of outputs from a set of inputs as a *forward model*. Once trained, *forward models* can readily predict outputs from their corresponding inputs. Sometimes, as is the case in this study, we may be interested to predict *inputs* from *outputs*. Assuming that the output layer is smaller in dimensionality than the input layer (which is true both in our study and in many neural network applications), then simply attempting to build a NN for the reverse mapping (i.e., predicting *inputs* from *outputs*) may result in poor model performance.

To address this, instead of building a NN to directly predict *inputs* (higher dimension) from *outputs* (lower dimension), we develop a *backward model*. This backward model leverages the forward model to efficiently search the input space for an input vector that produces a desired output vector. The model includes the same layers (and weights) as the forward model, which are fixed during training. Given a target *output* layer, the model performs gradient descent over its *input* layer so as to minimize the distance between the predicted and target output (as measured by mean squared error). The result is an input vector that (approximately) produces the desired output when fed through the forward model.

Since there may be many possible inputs for each output, the backward model will produce a locally-optimal weight solution (i.e., an input corresponding to the desired output) that can vary based on the input's initialization. Running the backward model multiple times, with different initializations, can thus yield multiple inputs that, when fed through the forward model, produce the desired output. More formally, if $h(\mathbf{x})$ represents the output of the learned forward model, $h(\cdot)$, given input $\mathbf{x}$, then for a desired output $\mathbf{y}$ the backward model (locally) solves $\arg\min_{\mathbf{x}} \|(\mathbf{y} - h(\mathbf{x})\|^2$, where $\|\cdot\|$ is the Euclidean norm.

Note that to make the search over the input space possible, the architecture of the backward model is slightly altered from the forward model. This is mainly to account for the fact that NNs do not perform gradient descent over their inputs (the desired operation for the backward model), but rather their weights. Specifically, we replace the forward model's input layer with a hidden layer, whose activations represent the input vector that is subsequently fed to the remaining layers. Preceding this hidden layer is a single input unit that is always set to 1, which is connected to the hidden layer via connection weights. The *weights* in this layer can thus be interpreted as the values of the *input* vector which is of interest. This is because setting their antecedent input unit to 1 ensures that their subsequent activations are equal to the weight values. This layer is the *only* layer that is trained during the backward model run, such that training the backward model to produce a desired output necessarily trains these weight values to be equal to a viable input solution. Thus, the backward model's 'solutions' are not neural network outputs, but rather the learned weights of the first hidden layer.

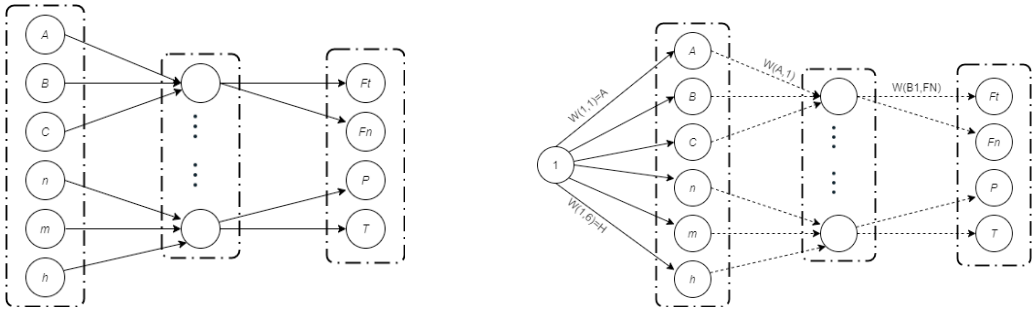
Note that given this architecture and training process, the backward model must be separately trained for each desired four-tuple output. This is because each training run produces a *single* weights vector for the first layer. Lastly, it is worth mentioning that this training architecture can yield multiple inputs for a given output (despite the model's fixed input unit of '1'). This is due to the initialization of the first layer's weights and the fact that, consistent with the literature, training stops when a local optimal is achieved.

2.3. Models

We leverage two NNs as described in Section 2.2: the *forward model*, predicts four-tuple output variables (F_t, F_n, P, T) from a given six-tuple input vector (five J-C parameters and chip thickness h) and the *backward model*, searches the input space of the forward model to estimate a six-tuple input vector that produces a target four-tuple output.

Forward NN model: We develop a multivariate NN regression model consisting of three layers, i.e., an input layer with six neurons for six-tuple inputs, a hidden layer with 20 neurons (the number of the neurons of the hidden layer is determined by back-propagation method [39]), and an output layer with four neurons for four-tuple outputs. Fig. 3(a) shows *forward NN model* contains one input layer with six neurons representing six-tuple input, one hidden layer with twenty neurons, and an output layer with four neurons representing four-tuple output.

The *forward NN model* uses a uniform kernel initializer with a random seed. The rectified linear activation function (RELU) is used as the activation function in the hidden layer. The linear activation function is used for the output layer.



(a) Forward NN model (b) Backward NN model

Fig. 3: (a) *Forward NN model* consists of one input layer with six neurons representing six-tuple input, one hidden layer with 20 neurons, and an output layer with four neurons representing four-tuple output; (b) *Backward NN model* consists of one input layer of a single neuron, two hidden layers, and an output layer with four neurons. In essence, this model includes the trained forward model, plus one additional layer that is trainable. Specifically, the solid lines indicate trainable parameters, while dashed lines indicate previously trained/fixed parameters. The model is trained on the input layer only so it learns to adjust the six-tuple weights of the input layer to predict the target four-tuple output. The resulting adjusted weights are the six-tuple input variables of interest.

Backward NN model: As described in Section 2.2, the *backward model* reformulates the six-tuple input layer of the forward model as a hidden layer, then performs gradient descent over this layer to yield the six-tuple input that corresponds to a desired four-tuple output. Fig. 3(b) shows *backward model* with solid lines indicate trained parameters, dashed lines indicate fixed parameters. The backward NN model uses a constant kernel initializer and zero bias with a sigmoid activation function in the input layer. All models are developed in python using the library Keras [40].

2.4. Model Training and Evaluation

We set aside 30 percent of the data for testing. In the *forward model*, we use 70 percent of the remaining data for training and 30 percent for validation. In training, the *forward model* uses a batch size of 50 and trains for 250 epochs with an early stopping based on validation loss, and a patience of 15. Recall that the *backward model* is a modified neural network method that searches the *forward model's* input space to identify an input vector that maps to a desired output vector. Hence, because the adjusted weights of the input layer are the estimated J-C parameters and chip thickness, no data are separated for testing.

To evaluate the forward and backward models, we perform a series of steps. Next, we provide an overview of these steps (also depicted in Figure 4) and introduce our notation as follows. We then provide additional details.

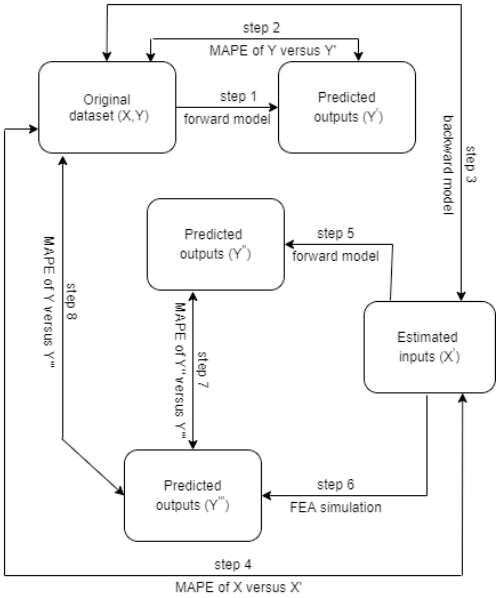


Fig. 4: Overview of the evaluation process.

Step 0: We denote the original dataset by (X, Y), with the input variables X (i.e., six-tuple J-C parameters and chip thickness) and the output variables Y (i.e., four-tuple (F_t, F_n, P, T)'s).

Step 1: Use the *forward model* to predict four-tuple output variables, yielding predictions Y'.

Step 2: Compare Y and Y'. This provides an estimated error between ground truths Y and the predicted output variables Y'.

Step 3: Given output variables Y, use the *backward model* to estimate input variables X'. We execute the model using initial random weights with up to 1%, 5%, 10%, 15%, 20%, and 25% deviations from ground truths X. These deviations are equivalent to making 'educated guesses' of the ground truth input values that yield Y.

Step 4: Compare X and X'. This provides an estimated error between ground truths X and the estimated input variables X'.

Step 5: Use X' in the *forward model* to predict four-tuple output variables, yielding predictions Y''.

Step 6: Use X' in the FEA software to perform simulations and obtain four-tuple output variables Y'''.

Step 7: Compare Y'' and Y'''. This provides an estimated error between the predicted four-tuple output variables from the *forward model* Y'' and the FEA simulated four-tuple outputs Y'''.

Step 8: Compare Y and Y'''. This provides an estimated error between the ground truths Y and the FEA simulated four-tuple outputs Y'''.

For *forward model* evaluation (evaluation steps 1-2), we use mean absolute percentage error (MAPE) which measures the percentage error between the predicted output variables and the ground truths (i.e., actual variable values). Furthermore, to present the degree of generalizability of the model, we perform bootstrapping 10 times [41] per the 30%/70% training and testing breakdown, and evaluate average MAPE across the trials and provide confidence intervals (CIs) [42]. Finally, we benchmark these results against those of univariate regression models. That is, we show the discrepancy in MAPE when the *forward model* applies univariate regression, predicting each response variable separately, versus multivariate regression, predicting all response variable simultaneously, and provide insights about the observed discrepancy.

For *backward model* evaluation (evaluation steps 3-8), we also use MAPE. First (evaluation steps 3-4), for any given four-tuple output, we assess the backward model's ability to recover the corresponding six-tuple input vector. To do this, we initialize the *backward model*'s first layer with weights that differ from the ground truth six-tuple input vector by varying amounts (1-25% deviations as well as random initialization). We then calculate MAPE between the ground truth and the estimated six-tuple inputs. This allows us to assess the model's ability to recover six-tuple ground truth inputs even when its initialization differs considerably from these desired six-tuple vectors.

Note that since there may be multiple six-tuple inputs that produce the same four-tuple output, the backward model may yield different six-tuple input vectors than the ground truths. Thus (evaluation steps 5-8), we evaluate whether the backward model's recovered six-tuple vectors produce the desired four-tuple outputs, *even if the input vectors differ from the ground truths in the training set*. As such, we first use the estimated six-tuple input vectors and compare the resulting predicted four-tuple output variables from the *forward model* and the FEA software to validate the applicability of the *forward model* to these data. Next, we compare the four-tuple ground truth outputs in the training set with the simulated outputs that are obtained from the estimated six-tuple input vectors. This evaluates the backward model's ability to identify other 'candidate' six-tuple inputs for a desired four-tuple output, besides those in the training set.

3. Numerical Results

3.1. Descriptive Statistics

Table 2 provides the mean, standard deviation (SD), and quantiles of the variables included in this study. Fig. 5 shows the histograms of the four FEA simulation output variables. As seen in the figure, despite the relatively skewed/non-uniform input variables, the FEA simulation output variables are approximately normally distributed.

Table 2: Descriptive statistics of the variables.

	Mean	**SD**	**Min**	**25%**	**50%**	**75%**	**Max**
A (MPa)	282.38	27.21	164.00	282.94	282.94	282.94	378.63
B (MPa)	107.12	23.55	17.87	109.10	109.10	109.10	211.00
C	0.0096	0.0056	0.0010	0.0081	0.0081	0.0081	0.0411
n	0.3839	0.0699	0.0840	0.3905	0.3905	0.3905	0.6606
m	1.3150	0.0860	0.8239	1.3208	1.3208	1.3208	1.5840
h (mm)	0.1500	0.0141	0.1300	0.1400	0.1500	0.1600	0.1700
F_t (N)	130.84	14.30	89.72	121.07	130.32	139.41	193.33
F_n (N)	85.73	8.79	60.79	80.02	85.29	90.45	125.42
P (W)	1046.67	114.32	717.73	968.58	1042.51	1115.31	1546.60
T (°C)	320.55	25.43	241.56	308.28	321.45	331.95	424.87

3.2. Forward model

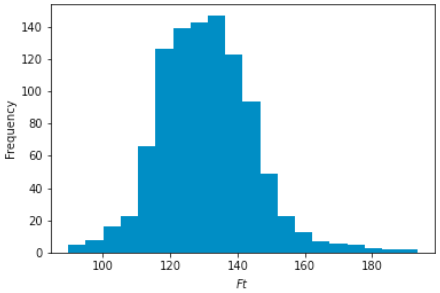
Table 3 presents the MAPE and the corresponding 95% confidence interval (CI), per evaluation step 2, for the 10 bootstrapped trials when forward model performs multivariate regression, predicting all four-tuple outputs simultaneously. Compare these results with those in Table 4, which presents the MAPE and the corresponding 95% CI for the 10 bootstrapped trials when forward model performs univariate regression, predicting each output separately. As seen in the tables, the MAPE values are overall lower when performing multivariate regression. Specifically, the mean MAPE for multivariate forward model is 2.085%, whereas the mean MAPE for the univariate forward model is larger, at 2.16%. This is due to the fact that the univariate forward model separately predicts each output variable, ignoring the correlation between the output values that is shown in Table 5.

Table 3: MAPE and CIs for *mutivariate forward model*.

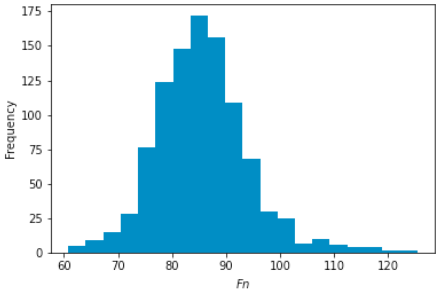
Variable	**MAPE (%)**	**95% confidence interval of MAPE**
F_t (N)	1.83	(1.59 , 2.06)
F_n (N)	2.32	(1.98 , 2.67)
P (W)	1.89	(1.65 , 2.14)
T (°C)	2.30	(1.98 , 2.63)

3.3. Backward model

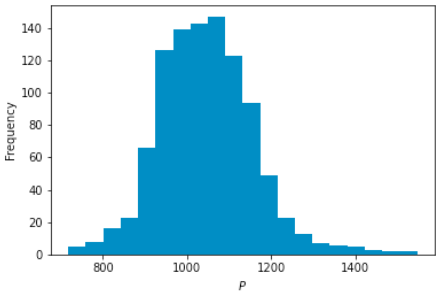
Table 6 presents an example of the evaluation step 4, where we compare the ground truths versus estimated values from the *backward model* for a randomly selected example from the dataset (with the target values of F_t = 125.4330 (N), F_n = 83.9820 (N), P = 1003.4700 (W), T = 322.1459 (°C) and initializers 1% deviated from the ground truths). As seen



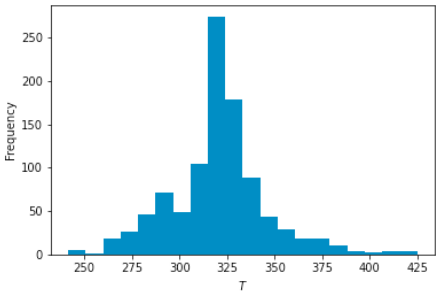
(a) Histogram of F_t (N)



(b) Histogram of F_n (N)



(c) Histogram of P (W)



(d) Histogram of T (deg C)

Fig. 5: Histograms of the output variables of FEA simulated dataset. All output variables are approximately normally distributed.

Table 4: MAPE and CIs for *univariate forward model*.

Variable	**MAPE (%)**	**95% confidence interval of MAPE**
F_t (N)	1.98	(1.74 , 2.22)
F_n (N)	2.11	(1.70 , 2.51)
P (W)	2.04	(1.71 , 2.36)
T (°C)	2.51	(2.14 , 2.89)

Table 5: Correlation matrix for FEA output variables.

	F_t (N)	F_n (N)	P (W)	T (deg C)
F_t (N)	1.0000	0.7350	1.0000	0.7024
F_n (N)	0.7350	1.0000	0.7350	0.8479
P (W)	1.0000	0.7350	1.0000	0.7024
T (°C)	0.7024	0.8479	0.7024	1.0000

in the table, the absolute percentage deviation (APD) of the estimated values from target values is very low, suggesting that the backward model can estimate the J-C parameters and chip thickness within 99% accuracy starting with a small deviation for this randomly selected example.

Next, we feed these estimated values to the *forward model* to examine if these estimated values can indeed recover the desired four-tuple outputs. Table 7 presents the ground truth values of the FEA output variables from the dataset that are used to estimate the input variables given in Table 6, compared with their predicted values by the *forward model*. As seen in the table, as expected, the ground truths are very close to the predicted values.

Table 6: Ground truths versus estimated values by the *backward model* for a randomly selected example from the dataset with the target values of F_t = 125.4330, F_n = 83.9820, P = 1003.4700, T = 322.1459 and initializers 1% deviated from the ground truths. [APD: absolute percentage deviation]

Variables	**Ground Truths**	**Estimated Values**	**APD (%)**
A (MPa)	282.9391	283.4605	0.1842
B (MPa)	109.0947	109.2509	0.1431
C	0.0081	0.0080	1.2345
n	0.3905	0.3930	0.6402
m	1.3208	1.3172	0.2725
h (mm)	0.1400	0.1399	0.0714

Table 7: Ground truths versus predicted values by the *forward model* for the ground truths and estimated variables presented in Table 6.

Variables	**Ground Truths**	**Predicted Values**
F_t (N)	125.4330	124.9237
F_n (N)	83.9820	82.0941
P (W)	1003.4700	999.1153
T (°C)	322.1450	318.5349

Table 8: Mean MAPE between the ground truths and the estimated input variables from the *backward model*, for 10 random examples at various degrees of deviations in weight initializations.

Deviation (%)	A (MPa)	B (MPa)	C	n	m	h (mm)
1	0.0778	0.3454	0.5407	0.2395	0.21226	0.0272
5	0.5306	1.4752	4.1146	1.6229	1.4632	0.1475
10	1.6122	3.1445	7.3996	5.1117	2.5431	0.2945
15	2.3090	7.2557	11.9730	5.0682	5.0269	0.6133
20	3.3216	9.8737	13.6400	13.1130	5.1116	0.7455
25	3.2894	16.3770	14.4090	14.4090	6.8646	0.6105

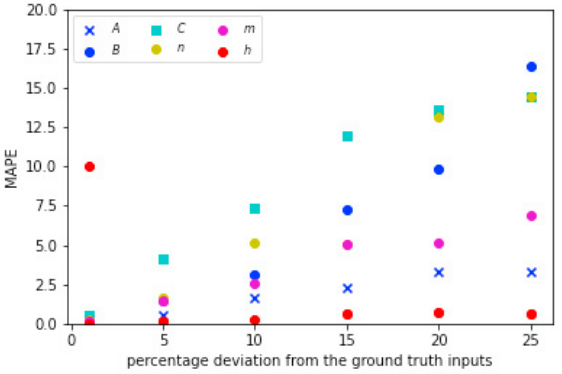


Fig. 6: Mean MAPE between the ground truths and the estimated input variables from the *backward model*, for 10 random examples at various degrees of deviations in weight initializations.

In the following, we more thoroughly compare the ground truth inputs and the estimated inputs, as well as the ground truth outputs and the predicted and simulated outputs from the estimated inputs for a larger set of examples. Table 8 shows the mean MAPE between the ground truths and the estimated input variables (per evaluation steps 3-4), for 10 random examples given the various degrees of deviations in weight initializations. As Table 8 suggests, the backward model can generally estimate input variables that are very close to the ground truths for small deviations in weight initializations (i.e., 1-5%). However, the errors are somewhat larger for larger deviations (i.e., 20-25%), with up to 16% error when starting as far as 25% away from ground truths. Fig. 6 presents the scatter plot of Table 8.

Next, per evaluation steps 5-8, we investigate whether the estimated input variables, fed to the *forward model*, will result in predicted four-tuple output variables that are consistent with the results of FEA software, and whether these resulting FEA simulated outputs are close to the ground turth outputs in the training set. As discussed, if this is the case, it suggests that alternative, viable 'candidate' solutions are identified by the NN-based approach. Table 9 presents the mean MAPE between the predicted four-tuple output variables from the *forward model* and the FEA simulated outputs for the estimated input values. As seen in the table, the mean MAPE across all variables is

Table 9: Mean MAPE between predicted four-tuple output variables from the *forward model* and the FEA simulated outputs for the estimated input values, whose difference with ground truth inputs are presented in Table 8.

Deviation %	F_t (%)	F_n (%)	P (%)	T (%)	Average (%)
1	1.94	1.31	2.28	1.37	1.73
5	1.99	1.30	2.33	1.57	1.80
10	0.40	1.31	0.40	0.56	0.67
15	2.06	1.14	2.33	1.47	1.75
20	2.08	1.54	2.12	1.50	1.81
25	2.52	1.82	2.91	1.44	2.17

limited to 3%, suggesting that the *forward model* can be confidently applied to the estimated inputs and will produce results that are comparable to those of an FEA software. Next, Table 10 presents the mean MAPE between the ground truths and the FEA simulated outputs for the estimated input values. Again, as seen in the table, the mean MAPE across all variables is limited to 5%, whereas the mean MAPE in the estimated input values ranges up to 16% in Table 8. These results suggest that although the estimated input values may be different from those in the training set, they indeed produce the the four-tuple output values of interest. Hence, in each execution, the proposed NN-based approach can properly estimate one of the many sets of J-C parameters and chip thickness that would result in the target output variable.

Table 10: Mean MAPE between the ground truths and and FEA simulated outputs for the estimated input values, whose difference with ground truth inputs are presented in Table 8.

Deviation %	F_t (%)	F_n (%)	P (%)	T (%)	Average (%)
1	0.29	0.94	0.29	0.32	0.45
5	0.50	0.96	0.51	0.47	0.61
10	0.51	0.96	0.51	.47	0.67
15	0.40	1.31	0.40	0.56	0.54
20	4.51	4.94	4.51	4.05	4.50
25	1.03	1.50	1.03	1.01	1.14

4. Discussion and Future Work

The goal of this study was to develop an approach to estimate J-C parameters and chip thickness for a given set of outputs including cutting force, stress, strain, and temperature, using a dataset obtained form FEA software simulations. The task at hand was complicated due to a many-to-one relationship where different sets of J-C parameters and chip thickness resulted in similar outputs. We developed a NN-based approach to estimate the parameters despite such many-to-one relationships. Specifically, we developed a *forward model*, as a function approximator, to bypass the need for future simulations. In addition, we used this model as part of a *backward model* to efficiently search the input space of the *forward model* to estimate J-C parameters and chip thickness that would result in a targeted output vector of interest.

We performed throughout analysis and evaluation of the proposed approach. Our results showed that even when initializing the weights in the *backward model* far away from their ground truths, the model was able to recover viable 'candidate' J-C parameters and chip thickness. Specifically, even with up to 25% deviations from the ground truths, which resulted in up to 16% MAPE between the ground truth inputs and the estimated inputs (Fig. 6 and Table 8), the estimated inputs indeed pointed to the targeted outputs of interests, with only up to 5% MAPE (Table 10).

Because most of the existing studies estimate J-C parameters using experimental methods such as SHPB tests and the detailed values are not readily accessible, we leave thorough benchmarking of the proposed approach to future work. As a straightforward benchmark for the proposed NN-based approach, and in particular the *backward model*, however, we developed a new NN model, which simply mapped the four-tuple outputs to the six-tuple inputs. Per our preliminary results, the average MAPE of this NN model was on the order of 12%, which is much higher than the up to 5% MAPE reported in Section 3.

It is worth mentioning that in the current experiments, we initialize the weights in the *backward model* that are relatively close to the input ground truths, i.e., randomly selected values with up to 25% deviation. This is equivalent to an 'educated guess' by the user. Therefore, the next steps include extending the model to be able to estimate the input variables starting from any initial random weights.

In addition, future research includes incorporating generative models, such as variational autoencoders [43, 44] or generative adversarial networks [45], into the backward model architecture, which can generate a distribution of feasible input parameters given a desired output. Another extension is to modify the approach to estimate multiple candidate inputs (multiple local optimal solutions) in a single execution of the model.

5. Conclusion

In this study, we developed an NN-based approach to estimate six-tuple J-C parameters and chip thickness, given a targeted set of four-tuple outputs such as cutting force, stress, strain, and temperature. A straightforward NN model was unable to predict the six-tuple inputs from the four-tuple outputs. Hence, we devised an unorthodox *backward model* to search the input space of a *forward model* to accurately estimate the six-tuple inputs. Our results suggest that the proposed approach can estimate the J-C parameters and and chip thickness such that when they are implemented as part of FEA simulations, they can closely produce the targeted outputs.

6. Appendix

Dataset is available at this link.
Code is available at GitHub.

References

[1] An He, Ganlin Xie, Hailong Zhang, and Xitao Wang. A comparative study on johnson–cook, modified johnson–cook and arrhenius-type constitutive models to predict the high temperature flow stress in 20crmo alloy steel. *Materials Design (1980-2015)*, 52:677 – 685, 2013.

[2] F. Ducobu, E. Rivière-Lorphèvre, and E. Filippi. On the importance of the choice of the parameters of the johnson-cook constitutive model and their influence on the results of a ti6al4v orthogonal cutting model. *International Journal of Mechanical Sciences*, 122:143 – 155, 2017.

[3] A. Dorogoy and D. Rittel. Determination of the johnson–cook material parameters using the scs specimen. *Experimental Mechanics*, 49(6):881, Nov 2008.

[4] Akhtar S. Khan, Yeong Sung Suh, and Rehan Kazmi. Quasi-static and dynamic loading responses and constitutive modeling of titanium alloys. *International Journal of Plasticity*, 20(12):2233 – 2248, 2004.

[5] H Kolsky. An investigation of the mechanical properties of materials at very high rates of loading. *Proceedings of the Physical Society. Section B*, 62(11):676–700, nov 1949.

[6] Jinqiang Ning and Steven Y. Liang. Model-driven determination of johnson-cook material constants using temperature and force measurements. *The International Journal of Advanced Manufacturing Technology*, 97(1):1053–1060, Jul 2018.

[7] Jinqiang Ning, Vinh Nguyen, Yong Huang, Karl T. Hartwig, and Steven Y. Liang. Inverse determination of johnson–cook model constants of ultra-fine-grained titanium based on chip formation model and iterative gradient search. *The International Journal of Advanced Manufacturing Technology*, 99(5):1131–1140, Nov 2018.

[8] F. Klocke, D. Lung, and S. Buchkremer. Inverse identification of the constitutive equation of inconel 718 and aisi 1045 from fe machining simulations. *Procedia CIRP*, 8:212 – 217, 2013. 14th CIRP Conference on Modeling of Machining Operations (CIRP CMMO).

[9] P Changizian, A Zarei-Hanzaki, and Ali A Roostaei. The high temperature flow behavior modeling of az81 magnesium alloy considering strain effects. *Materials & Design*, 39:384–389, 2012.

[10] P. Changizian, A. Zarei-Hanzaki, and Ali A. Roostaei. The high temperature flow behavior modeling of az81 magnesium alloy considering strain effects. *Materials Design*, 39:384 – 389, 2012.

[11] Yin-Jiang Qin, Qing-Lin Pan, Yun-Bin He, Wen-Bin Li, Xiao-Yan Liu, and Xi Fan. Modeling of flow stress for magnesium alloy during hot deformation. *Materials Science and Engineering: A*, 527(10-11):2790–2797, 2010.

[12] Bin Shi, Helmi Attia, and Nejah Tounsi. Identification of material constitutive laws for machining—part i: an analytical model describing the stress, strain, strain rate, and temperature fields in the primary shear zone in orthogonal metal cutting. *Journal of Manufacturing Science and Engineering*, 132(5), 2010.

[13] J Pujana, PJ Arrazola, R M'saoubi, and H Chandrasekaran. Analysis of the inverse identification of constitutive equations applied in orthogonal cutting process. *International Journal of Machine Tools and Manufacture*, 47(14):2153–2161, 2007.

[14] Aviral Shrot and Martin Bäker. Determination of johnson–cook parameters from machining simulations. *Computational Materials Science*, 52(1):298 – 304, 2012. Proceedings of the 20th International Workshop on Computational Mechanics of Materials - IWCMM 20.

[15] Aviral Shrot and Martin Bäker. Is it possible to identify johnson-cook law parameters from machining simulations? *International Journal of Material Forming*, 3(1):443–446, Apr 2010.

[16] A.S. Milani, W. Dabboussi, J.A. Nemes, and R.C. Abeyaratne. An improved multi-objective identification of johnson–cook material parameters. *International Journal of Impact Engineering*, 36(2):294 – 302, 2009.

[17] G.H. Majzoobi, F. Freshteh-Saniee, S. Faraj Zadeh Khosroshahi, and H. Beik Mohammadloo. Determination of materials parameters under dynamic loading. part i: Experiments and simulations. *Computational Materials Science*, 49(2):192 – 200, 2010.

[18] G.H. Majzoobi, S. Faraj Zadeh Khosroshahi, and H. Beik Mohammadloo. Determination of materials parameters under dynamic loading: Part ii: Optimization. *Computational Materials Science*, 49(2):201 – 208, 2010.

[19] Dong-Hyeon Kim, Thomas J. Y. Kim, Xinlin Wang, Mincheol Kim, Ying-Jun Quan, Jin Woo Oh, Soo-Hong Min, Hyungjung Kim, Binayak Bhandari, Insoon Yang, and Sung-Hoon Ahn. Smart machining process using machine learning: A review and perspective on machining industry. *International Journal of Precision Engineering and Manufacturing-Green Technology*, 5(4):555–568, Aug 2018.

[20] Kai Yang and Jayant Trewn. *Multivariate statistical methods in quality management*. McGraw Hill Professional, 2004.

[21] Gülser Köksal, İnci Batmaz, and Murat Caner Testik. A review of data mining applications for quality improvement in manufacturing industry. *Expert systems with Applications*, 38(10):13448–13467, 2011.

[22] Ravindranadh Bobbili, B. Ramakrishna, V. Madhu, and A.K. Gogia. Prediction of flow stress of 7017 aluminium alloy under high strain rate compression at elevated temperatures. *Defence Technology*, 11(1):93 – 98, 2015.

[23] N. Haghdadi, A. Zarei-Hanzaki, A.R. Khalesian, and H.R. Abedi. Artificial neural network modeling to predict the hot deformation behavior of an a356 aluminum alloy. *Materials Design*, 49:386 – 391, 2013.

[24] Issam S Jalham. Modeling capability of the artificial neural network (ann) to predict the effect of the hot deformation parameters on the strength of al-base metal matrix composites. *Composites Science and Technology*, 63(1):63 – 67, 2003.

[25] Salim Heddam, Behrooz Keshtegar, and Ozgur Kisi. Predicting total dissolved gas concentration on a daily scale using kriging interpolation, response surface method and artificial neural network: Case study of columbia river basin dams, usa. *Natural Resources Research*, 29(3):1801–1818, Jun 2020.

[26] Rahul R. Chakule, Sharad S. Chaudhari, and P. S. Talmale. Evaluation of the effects of machining parameters on mql based surface grinding process using response surface methodology. *Journal of Mechanical Science and Technology*, 31(8):3907–3916, Aug 2017.

[27] Asit Kumar Parida and Kalipada Maity. Modeling of machining parameters affecting flank wear and surface roughness in hot turning of monel-400 using response surface methodology (rsm). *Measurement*, 137:375 – 381, 2019.

[28] Amir Malakizadi, Stefan Cedergren, Ibrahim Sadik, and Lars Nyborg. Inverse identification of flow stress in metal cutting process using response surface methodology. *Simulation Modelling Practice and Theory*, 60:40–53, 2016.

[29] Sohail Akram, Syed Husain Imran Jaffery, Mushtaq Khan, Muhammad Fahad, Aamir Mubashar, and Liaqat Ali. Numerical and experimental investigation of johnson–cook material models for aluminum (al 6061-t6) alloy using orthogonal machining approach. *Advances in Mechanical Engineering*, 10(9):1687814018797794, 2018.

[30] Amir H Adibi-Sedeh, Vis Madhavan, and Behnam Bahr. Extension of oxley's analysis of machining to use different material models. *J. Manuf. Sci. Eng.*, 125(4):656–666, 2003.

[31] IS Boldyrev, IA Shchurov, and AV Nikonov. Numerical simulation of the aluminum 6061-t6 cutting and the effect of the constitutive material model and failure criteria on cutting forces' prediction. *Procedia Engineering*, 150:866–870, 2016.

[32] M Daoud, W Jomaa, JF Chatelain, and A Bouzid. A machining-based methodology to identify material constitutive law for finite element simulation. *The International Journal of Advanced Manufacturing Technology*, 77(9-12):2019–2033, 2015.

[33] M Daoud, W Jomaa, Jean-François Chatelain, A Bouzid, and Victor Songmene. Identification of material constitutive law constants using machining tests: a response surface methodology based approach. *WIT Transactions on The Built Environment*, 137:25–36, 2014.

[34] N Fang. A new quantitative sensitivity analysis of the flow stress of 18 engineering materials in machining. *J. Eng. Mater. Technol.*, 127(2):192–196, 2005.

[35] Patxi Fernandez-Zelaia and Shreyes N Melkote. Statistical calibration and uncertainty quantification of complex machining computer models. *International Journal of Machine Tools and Manufacture*, 136:45–61, 2019.

[36] Gaetano Massimo Pittalà and Michele Monno. 3d finite element modeling of face milling of continuous chip material. *The International Journal of Advanced Manufacturing Technology*, 47(5-8):543–555, 2010.

[37] William Keith Rule. A numerical scheme for extracting strength model coefficients from taylor test data. *International journal of impact engineering*, 19(9-10):797–810, 1997.

[38] Imed Zaghbani and Victor Songmene. A force-temperature model including a constitutive law for dry high speed milling of aluminium alloys. *Journal of Materials Processing Technology*, 209(5):2532–2544, 2009.

[39] Kazuhiro Shin-ike. A two phase method for determining the number of neurons in the hidden layer of a 3-layer neural network. In *Proceedings of SICE Annual Conference 2010*, pages 238–242. IEEE, 2010.

[40] Antonio Gulli and Sujit Pal. *Deep learning with Keras*. Packt Publishing Ltd, 2017.

[41] Andreas S Weigend, Blake LeBaron, et al. Evaluating neural network predictors by bootstrapping. In *Proc. Int. Conf. Neural Inform. Processing*, volume 2, pages 1207–1212. Citeseer, 1994.

[42] Thomas J DiCiccio and Bradley Efron. Bootstrap confidence intervals. *Statistical science*, pages 189–212, 1996.

[43] D.P. Kingma and M. Welling. Auto-encoding variational Bayes. *arXiv preprint arXiv:1312.6114*, 2013.

[44] Mihaela Rosca, Balaji Lakshminarayanan, David Warde-Farley, and Shakir Mohamed. Variational approaches for auto-encoding generative adversarial networks. *arXiv preprint arXiv:1706.04987*, 2017.

[45] I. Goodfellow, J. Pouget-Avadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014.