MACHINE LOAD ESTIMATION VIA STACKED AUTOENCODER REGRESSION

Tulika Bose⋆ Angshul Majumdar† Tanushyam Chattopadhyay⋆

⋆ TCS Research and Innovation, Tata Consultancy Services, Kolkata, India
† Indraprastha Institute of Information Technology, Delhi, India

ABSTRACT

The problem of load estimation from sensor signals holds significance in the field of intelligent manufacturing. The goal of this work is to estimate the axial and spindle load values in a Computer Numerical Control machine from input sensor readings like spindle speed, feed rate, tool positions, etc. This can be viewed as a standard regression problem. Here, we propose a novel deep learning based regression technique that incorporates regression within the stacked autoencoder framework. Unlike the popular heuristic pre-training, fine-tuning approach, we solve all the parameters of the problem jointly. A variable splitting Augmented Lagrangian approach is employed to solve the ensuing optimization problem. Comparisons on standard regression models like linear, Least Absolute Shrinkage and Selection Operator (LASSO), Support Vector Regression and the traditional stacked autoencoder have shown that our technique considerably outperforms them.

Index Terms—Asymmetric stacking, deep learning, regression, estimation, intelligent manufacturing

1. INTRODUCTION

The intelligent manufacturing domain has been automated to a great extent with the introduction of the Computer Numerical Control (CNC) machines [1]. The entire machining process involves complex interaction of several operating signals obtained from different sensors [2]. Since the tool surfaces undergo continuous wearing with the progress of the operation and often result in tool breakage, tools are replaced manually in a timely manner based on simple rules like the number of workpieces completed and the like [3]. However, in order to make the tool replacement strategy cost-effective as well as to prevent damage due to tool breakage, this process needs to be automated.

The load or power incurred across the axes and the spindle during the actual cutting process are effective in detecting breakage of the cutting tool [4]. Thus the focus of our paper is to mathematically model and estimate the axial load and the spindle load using sensor signals associated with the machine, including feed rate, surface speed, spindle speed, etc., for better productivity and process control [5, 6].

Although ideally a physics based model can be built for the above problem, it is not always possible to develop them due to the complicated dynamics of manufacturing systems. In this work we will go for a completely data driven approach and formulate it as a regression problem. We propose a deep learning architecture based on the stacked autoencoder framework to solve this regression problem as it can learn arbitrary relationships without specification from the user. For such supervised deep learning problems, the standard approach is to follow the pre-training and fine-tuning paradigm [7]. In the pre-training stage, each layer of the stacked autoencoder is learnt greedily. In the fine-tuning stage, the targets are attached at the deepest layer of the encoder and the decoder is detached. The supervised architecture is fine tuned. However, in such a greedy paradigm, there is no relationship between the parameters learnt during the pre-training and the fine-tuning stages.

In this work, we propose to incorporate regression into the stacked autoencoder framework. But instead of learning it greedily, we jointly solve for all the weights in a mathematically more optimal fashion, without using backpropagation. This will be a novel formulation, which will be solved using variable splitting Augmented Lagrangian techniques [8, 9]. The remainder of the paper is structured in the following manner. Section 2 gives a brief review of the prior literature. The proposed regression model is explained in detail in Section 3. This is followed by experimental evaluation and results in Section 4. Finally Section 5 concludes our work.

2. LITERATURE REVIEW

2.1. Regression for machine automation

There exist several prior-arts for regression models, amongst which some of the popular ones are the linear regression, Generalised Linear Models (GLM) [10], quasi-likelihood estimation [11], Least Absolute Shrinkage and Selection Operator (LASSO) [12], non-parametric Nadaraya-Watson kernel estimator [13,14] and Support Vector Regression (SVR) [15]. However, since only few of these regression models have been used for the problems of load estimation and machine automation and also due to the limitation of space, we shall refrain from discussing all of these models here.
Linear regression models have been explored in the intelligent manufacturing domain for various problems like spindle load estimation [16], error compensation [17] and the like. Given the load \( y_t \) and the signal predictors \( x_i = \{x_{1i}, \ldots, x_{ip}\}_{i=1}^n \) the model is given by \( y_t = x_i^T \beta + \epsilon_i \). Here \( \beta \) is the parameter vector and \( \epsilon \) is the error term or noise.

Non-linear regression techniques, especially SVR with polynomial kernels and gaussian kernels, have been used widely in machine automation use-cases. SVRs with these kernels have been applied for tool condition assessment in CNC machine [18], assessment of machine degradation [19], etc.

Besides, feature extraction and training Artificial Neural Networks (ANN) for regression have been performed for load estimation [16], tool wear prediction [20] and the like.

### 2.2. Autoencoders

Autoencoders, as shown in Fig. 1, are self-supervised neural networks, i.e. the inputs and the outputs are the same. The input data (\( X \) formed by stacking the training samples as columns of a matrix) is projected onto the hidden representation: \( H = \varphi(W_E X) \) by an encoder \( W_E \); there is a non-linear activation function \( \varphi \) associated with it. The decoder \( W_D \) reverse maps the representation onto the output (=input) as \( X = W_D H \). During training, the encoder and the decoder are learnt by minimizing the Euclidean cost function:

\[
\arg \min_{W_E, W_D} \| X - W_D \varphi(W_E X) \|_F^2 
\]

Stacked autoencoders are created by nesting autoencoders one inside the other. Mathematically this is expressed as

\[
\arg \min_{W_{E_1}, W_{E_2}, \ldots, W_{E_L}, W_D_1, W_D_2, \ldots, W_D_L} \| X - g(f(X)) \|_F^2
\]

where \( g(f(X)) = W_{D_1} \varphi(W_{D_2} \varphi(W_{D_3} \varphi(\ldots(W_{D_L} f(X))))) \) and \( f(X) = \varphi(W_{E_L} \varphi(W_{E_{L-1}} \varphi(\ldots(W_{E_1} X)))) \).

It is difficult to learn all the parameters in one go by backpropagation owing to the vanishing gradient problem. Therefore, the stacked autoencoder is solved greedily one layer at a time [7] starting from the outermost layer.

However, greedy learning is a sub-optimal approach since the outer layers influence the inner layers but not vice versa. In optimal learning, all the layers should be affecting one another (unlike the case when backpropagation is used).

### 3. OUR APPROACH

We propose a regression model based on stacked autoencoders with an asymmetric structure, consisting of multiple encoding layers and a single decoding layer. Previous studies on stacked autoencoders had equal number of encoders and decoders owing to the greedy training paradigm. Since we propose to learn all the layers jointly, we have the flexibility to change the architecture. We do not see any reason to have an equal number of encoders and decoders, especially since the decoders do not play any direct role in the analysis. More decoders mean more parameters to learn, which leads to over-fitting with limited training data. In a recent study [21], it was shown that having multiple encoders but only one decoder keeps the robust abstraction capacity of deep learning without the pitfalls of over-fitting. This motivates us to build upon the asymmetric architecture. We have shown the architecture for the two layer model, but one can easily extrapolate it to more layers.

![Fig. 1. Structure of an Autoencoder](image1)

![Fig. 2. Architecture of the proposed Asymmetric Stacked Autoencoder for Regression](image2)
structure that has two outputs. Thus the regression problem is addressed by solving the ensuing optimization problem. Here, a variable splitting Augmented Lagrangian paradigm is followed to solve the same [8, 9].

3.1. Joint-learning for regression

In contrast to the traditional two phase training model, the ensuing joint optimization problem for regression that needs to be solved in the training phase can be formulated as below:

$$\arg \min_{W_{E1}, W_{E2}, W_{D}, w} \left( \| X - W_{D} \varphi (W_{E2} \varphi (W_{E1}, X)) \|_{F}^{2} + \lambda \| y - w^{T} Z_{2} \|_{F}^{2} \right) \tag{3}$$

The symbols used here are described previously. Here $\lambda$ controls the weight of regression and $\varphi$ is the non-linear activation function. This is a non-convex joint optimization problem.

We introduce two proxy variables viz. $Z_{2} = \varphi (W_{E2} \varphi (W_{E1}, X))$ and $Z_{1} = \varphi (W_{E1}, X)$ [9]. The corresponding Augmented Lagrangian formulation is:

$$\arg \min_{W_{E1}, W_{E2}, W_{D}, w, Z_{2}, Z_{1}} \left( \| X - W_{D} Z_{2} \|_{F}^{2} + \lambda \| y - w^{T} Z_{2} \|_{F}^{2} + \mu_{2} \| Z_{2} - \varphi (W_{E2} Z_{1}) \|_{F}^{2} + \mu_{1} \| Z_{1} - \varphi (W_{E1}, X) \|_{F}^{2} \right) \tag{4}$$

where $Z_{1}$ and $Z_{2}$ are the representations of the input $X$ at the encoding layers. The parameter $\lambda$ and the hyper-parameters $\mu_{1}$ and $\mu_{2}$ need to be tuned as per the application. Breaking equation (4) into smaller pieces using Alternating Direction Method of Multipliers (ADMM) [22], we end up with 6 sub-problems as given below:

P1: $W_{D} \leftarrow \arg \min_{W_{D}} \| X - W_{D} Z_{2} \|_{F}^{2}$

P2: $w \leftarrow \arg \min_{w} \| y - w^{T} Z_{2} \|_{F}^{2}$

P3: $W_{E1} \leftarrow \arg \min_{W_{E1}} \| Z_{1} - \varphi (W_{E1}, X) \|_{F}^{2}$

$\Rightarrow W_{E1} \leftarrow \arg \min_{W_{E1}} \| \varphi^{-1} Z_{1} - W_{E1}, X \|_{F}^{2}$

P4: $W_{E2} \leftarrow \arg \min_{W_{E2}} \| Z_{2} - \varphi (W_{E2}, Z_{1}) \|_{F}^{2}$

$\Rightarrow W_{E2} \leftarrow \arg \min_{W_{E2}} \| \varphi^{-1} Z_{2} - W_{E2}, Z_{1} \|_{F}^{2}$

P5: $Z_{2} \leftarrow \arg \min_{Z_{2}} \left( \| X - W_{D} Z_{2} \|_{F}^{2} + \lambda \| y - w^{T} Z_{2} \|_{F}^{2} + \mu_{2} \| Z_{2} - \varphi (W_{E2} Z_{1}) \|_{F}^{2} \right)$

$\Rightarrow \arg \min_{Z_{2}} \left( \| \sqrt{\lambda} y \sqrt{\mu_{2}} \varphi (W_{E2} Z_{1}) \|_{F}^{2} \right) = \left( \sqrt{\lambda} w^{T} \sqrt{\mu_{2}} I \right) Z_{2}$

P6: $Z_{1} \leftarrow \arg \min_{Z_{1}} \left( \| W_{D} Z_{2} - \varphi (W_{E2} Z_{1}) \| F^{2} + \mu_{1} \| Z_{1} - \varphi (W_{E1}, X) \|_{F}^{2} \right)$

In contrast to the traditional two phase training model, the ensuing joint optimization problem for regression that needs to be solved in the training phase can be formulated as below:

Finally, during testing, the unknown output $\hat{y}$ for test data $X_{test}$ can be estimated using the learned weights by solving

$$\hat{y} = w^{T} \varphi (W_{E2} \varphi (W_{E1}, X_{test})) \tag{5}$$

4. EXPERIMENTAL EVALUATION

We have performed validation over smart factory real-world data involving 2 axis Horizontal and 4 axis Vertical CNC Turning machines. Since we have worked on proprietary data, we are unable to share the same publicly. The data consist of different sensor signals associated with the machine. The axial load and spindle load, incurred while machining specific components, have been measured in units of mega-watt (Mw) and recorded as ground truth. Data samples were collected per second throughout a day over a period of 1 month during the actual operating condition. The captured signals consist of absolute tool position across all the axes, amount of distance to go along the controlled axes, spindle speed value related to constant surface speed control on CNC, surface speed value, feed rate and the spindle motor speed. Our goal is to estimate load values at different axes as well as the spindle load using these captured signals.

<table>
<thead>
<tr>
<th>Axes names</th>
<th>$\lambda$</th>
<th>$\mu_{1}$</th>
<th>$\mu_{2}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Axis 1</td>
<td>1.6</td>
<td>1.3</td>
<td>1.3</td>
</tr>
<tr>
<td>Axis 2</td>
<td>0.5</td>
<td>1.7</td>
<td>1.7</td>
</tr>
<tr>
<td>Axis 3</td>
<td>0.6</td>
<td>1.6</td>
<td>1.6</td>
</tr>
<tr>
<td>Axis 4</td>
<td>1.1</td>
<td>1.5</td>
<td>1.5</td>
</tr>
<tr>
<td>Spindle</td>
<td>0.7</td>
<td>1.6</td>
<td>1.6</td>
</tr>
</tbody>
</table>

We have added non-linearity at each layer of the model using the hyperbolic tangent function. The number of nodes at hidden layers were adjusted experimentally. The proposed regression model was implemented in Matlab. The parameter values considered for our model, after performing grid search, are listed in Table 1. Table 2 presents the results obtained across 3 exemplary days using the proposed model. Though similar results were obtained for all the other days, they could
Table 2. Comparison of performance metrics using Linear, LASSO, SVR, traditional SAE and proposed regression model

<table>
<thead>
<tr>
<th>Day</th>
<th>Axis</th>
<th>Perf. metrics</th>
<th>Linear Regression</th>
<th>LASSO Regression</th>
<th>SVR (Polynomial)</th>
<th>SVR (Gaussian)</th>
<th>Traditional SAE</th>
<th>Proposed model</th>
</tr>
</thead>
<tbody>
<tr>
<td>4 Apr</td>
<td>Axis 3</td>
<td>RMSE</td>
<td>0.1208</td>
<td>0.1321</td>
<td>0.1307</td>
<td>0.1308</td>
<td>0.1320</td>
<td>0.1316</td>
</tr>
<tr>
<td></td>
<td></td>
<td>NMSE</td>
<td>5.0349</td>
<td>5.1313</td>
<td>7.0871</td>
<td>7.0864</td>
<td>5.0163</td>
<td>4.8417</td>
</tr>
<tr>
<td>4 Apr</td>
<td>Axis 4</td>
<td>RMSE</td>
<td>0.8018</td>
<td>0.8206</td>
<td>0.9593</td>
<td>0.9717</td>
<td>0.9800</td>
<td>0.6579</td>
</tr>
<tr>
<td>4 Apr</td>
<td>Spindle</td>
<td>RMSE</td>
<td>0.9406</td>
<td>0.9421</td>
<td>0.9955</td>
<td>0.9965</td>
<td>0.9745</td>
<td>0.8891</td>
</tr>
<tr>
<td>7 Apr</td>
<td>Axis 1</td>
<td>RMSE</td>
<td>0.6343</td>
<td>0.6353</td>
<td>0.6841</td>
<td>0.6852</td>
<td>0.7132</td>
<td>0.7132</td>
</tr>
<tr>
<td></td>
<td></td>
<td>NMSE</td>
<td>5.1879</td>
<td>5.1959</td>
<td>6.1761</td>
<td>6.1635</td>
<td>5.8331</td>
<td>4.4397</td>
</tr>
<tr>
<td>7 Apr</td>
<td>Axis 3</td>
<td>RMSE</td>
<td>0.1039</td>
<td>0.1042</td>
<td>0.1132</td>
<td>0.1147</td>
<td>0.1002</td>
<td>0.0980</td>
</tr>
<tr>
<td>7 Apr</td>
<td>Spindle</td>
<td>RMSE</td>
<td>0.8675</td>
<td>0.8806</td>
<td>0.9978</td>
<td>0.9990</td>
<td>0.9809</td>
<td>0.8141</td>
</tr>
<tr>
<td></td>
<td></td>
<td>NMSE</td>
<td>7.1641</td>
<td>7.2726</td>
<td>8.2515</td>
<td>8.2523</td>
<td>8.1010</td>
<td>6.7231</td>
</tr>
<tr>
<td>12 Apr</td>
<td>Axis 2</td>
<td>RMSE</td>
<td>0.6434</td>
<td>0.6477</td>
<td>0.7850</td>
<td>0.7857</td>
<td>0.5945</td>
<td>0.5587</td>
</tr>
<tr>
<td>12 Apr</td>
<td>Axis 3</td>
<td>RMSE</td>
<td>0.1082</td>
<td>0.1082</td>
<td>0.1168</td>
<td>0.1185</td>
<td>0.1026</td>
<td>0.1001</td>
</tr>
<tr>
<td>12 Apr</td>
<td>Axis 4</td>
<td>RMSE</td>
<td>0.7401</td>
<td>0.7591</td>
<td>0.9601</td>
<td>0.9806</td>
<td>0.9862</td>
<td>0.5372</td>
</tr>
</tbody>
</table>

Proposed model \((\text{nmse:0.1689,rmse:4.4333})\) and Linear Regression \((\text{nmse:0.1728, rmse:4.5357})\)

(a) Load in axis 1 on 17 Apr

(b) Load in axis 3 on 17 Apr

Fig. 3. Part of axial load values regressed using both linear regression and the proposed method.

5. CONCLUSION

In this work, we proposed a stacked autoencoder based novel regression framework. The focus of this paper is the problem of load estimation in a real time smart machine using captured sensor signals. We compared our regression framework with the existing popular methods like linear regression, LASSO, SVR as well as traditional stacked autoencoder and observed that our method outperforms them.

Although the proposed model is applied in the manufacturing domain, the same can be applied to any kind of regression problems, which makes the solution generic. For instance, the model can be applied on problems like prediction of household electricity consumption, blood pressure estimation using PPG and ECG signals and the like.
6. REFERENCES


