

# Pupil Learning Mechanism

Rua-Huan Tsaih, *Member, IEEE*, Yu-Hang Chien, *Member, IEEE*, and Shih-Yi Chien, *Member, IEEE*

**Abstract**—Studies on artificial neural networks rarely address both vanishing gradients and overfitting issues. In this study, we follow the pupil learning procedure, which has the features of interpreting, picking, understanding, cramming, and organizing, to derive the pupil learning mechanism (PLM) by which to modify the network structure and weights of 2-layer neural networks (2LNNs). The PLM consists of modules for sequential learning, adaptive learning, perfect learning, and less-overfitted learning. Based upon a copper price forecasting dataset, we conduct an experiment to validate the PLM module design modules, and an experiment to evaluate the performance of PLM. The empirical results indeed approve the PLM module design and show the superiority of the proposed PLM model over the linear regression model and the conventional backpropagation-based 2LNN model.

**Index Terms**—Computing methodologies, machine learning algorithms, artificial intelligence, adaptive neural networks, overfitting, pupil learning.

## I. INTRODUCTION

STUDIES of 2-layer neural networks (2LNNs) [1] rarely attempt to address both vanishing gradients and overfitting issues. In this study, we follow the pupil learning procedure shown in Table I to derive the pupil learning mechanism (PLM), which consists of modules for sequential learning, adaptive learning, perfect learning, and less-overfitted learning.

Many learning-based artificial intelligence (AI) studies deploy artificial neural networks (ANN) with a learning algorithm to a model, but encounter difficulties. For instance, backpropagation learning [1], which naively implements gradient descent optimization (GDO) to pursue weight estimates to minimize the loss function, encounters the difficulty of vanishing gradients. Furthermore, to progressively learn in a dynamic environment under a learning-based AI paradigm, the intelligence system should be self-adaptable, i.e., able to modify its own network structure and weights without human input. Sequential learning algorithms feature online processing, adjustable parameters, and an adaptable network structure [2]. Considerable efforts have been devoted to developing approaches for network structure adjustment, including (1) constructing approach: starting with a small network and then adding hidden nodes and connections [3], (2) pruning approach: starting with an extensive network and then removing irrelevant hidden nodes and connections [4]–[6], and (3) combining these constructing and pruning approaches [7]–[11]. However, these approaches fail to resolve overfitting challenges. An overfitted model is a model that (1) contains more parameters than can be justified by the data or (2) corresponds too closely or exactly with the training dataset and may therefore fail to fit testing data or predict future observations reliably [12], [13].

Recently, many learning-based AI studies deploy deep neural networks (DNN) with a deep learning (DL) algorithm that

TABLE I  
THE PUPIL LEARNING PROCEDURE THAT HAS THE FEATURES OF INTERPRETING, PICKING, UNDERSTANDING, CRAMMING, ORGANIZING (IPUCO)

- 
- When learning, the pupil quickly interprets instances, separating them into two groups: acquainted and unacquainted instances. Furthermore, the pupil typically learns unacquainted instances one by one and chooses easy (unacquainted) instances to learn first.
  - When encountering a new (unacquainted) instance, the pupil tries to understand it by using the learning process accompanied with his current knowledge. If it cannot be understood successfully, then the pupil crams it. The cramming process adds a strict rule to his knowledge system.
  - From time to time, the pupil comprehends all learned instances for a concise knowledge system.
- 

implements stochastic gradient descent optimization (SGDO) [14]. DNN models use complex network architectures that include many layers and nodes to learn highly nonlinear correlations among data. These DNNs have revolutionized the business and technology world with good performance in areas as varied as image classification, object detection and tracking, and video analytics. The frameworks of TensorFlow [15], [16] or PyTorch [17] and graphics processing units (GPUs) are used to enhance the learning performance of DL. However, complications still arise with DL, including long training times, overfitting, and the need for manual hyperparameter tuning [18], [19].

Various attempts [8], [9], [11] have been made to derive new learning approaches to resolve overfitting and vanishing gradients. For example, Tsaih [8], [9] adopts a hybrid approach to systematically construct and prune hidden nodes of 2LNN with the tanh activation function. [11] further revise this approach using the cramming, softening, and integrating (CSI) learning algorithm shown in Table II with parametric rectified linear unit (ReLU) activations instead of tanh activations. In addition, a variety of recent studies apply different methods to resolve the aforementioned issues [20]–[23]. However, most of the studies provide little empirical validation, making the proposed approaches controversial.

Extending [8], [9], [11] and following the pupil learning procedure shown in Table I, in this study we derive the PLM, which applies to 2LNNs with ReLU activations [24], [25] and contains modules for sequential learning, adaptive learning, perfect learning, and less-overfitted learning. Due to the variety of advanced modules developed to process data of high dimensions and large nonlinearity, it is mathematically infeasible to validate the proposed PLM. Therefore, in this study, we conduct a copper price forecasting experiment to empirically validate the PLM module design. We also conduct

TABLE II  
CSI (CRAMMING, SOFTENING, AND INTEGRATING) LEARNING  
ALGORITHM [11].

- Step 1: Use two reference observations  $\{(\mathbf{x}^1, y^1), (\mathbf{x}^2, y^2)\}$  with  $y^1 \cdot y^2 = -1$  to set up an acceptable SLFN estimate with one hidden node. Set  $n = 3$ .
- Step 2: If  $n > N$ , STOP.
- Step 3: Pick up the first  $n$  reference observations  $\{(\mathbf{x}^c, y^c)\}$  which are sorted by all  $N$  reference observations' squared residuals in ascending order. Let  $\mathbf{I}(n)$  be the set of indices of these picked observations.
- Step 4: If the condition  $L$  regarding  $\{f(\mathbf{x}^c, \mathbf{w}, r), \forall c \in \mathbf{I}(n)\}$  is satisfied, go to Step 7; otherwise, there is one and only  $\kappa \in \mathbf{I}(n)$  that is not at the right place.
- Step 5: Save  $\mathbf{w}$  and  $r$ .
- Step 6: Apply the weight-tuning mechanism to  $\min_{\mathbf{w}, r} E_n(\mathbf{w}, r)$  and adjust  $\mathbf{w}$  and  $r$  until one of the following two cases occurs:
- 1) If the condition  $L$  regarding  $\{f(\mathbf{x}^c, \mathbf{w}, r), \forall c \in \mathbf{I}(n)\}$  is satisfied, go to Step 7.
  - 2) If the condition  $L$  is not satisfied, restore  $\mathbf{w}$  and  $r$  then apply the cramming mechanism to add one extra hidden node to the existing SLFM to obtain an acceptable SLFN estimate.
- Step 7: Apply the softening and integrating mechanism to prune the irrelevant hidden node,  $n + 1 \rightarrow n$ ; go to Step 2.

a copper price forecasting experiment to empirically evaluate the performance of PLM. The empirical results indeed exhibit incremental learning, adaptive learning, perfect learning, and less-overfitted learning as well as that the PLM can effectively (1) adjusts the number of used hidden nodes according to the consequence of learning and new data, and (2) reduces the overfitting tendency while learning all training instances. The empirical results also show the superiority of the proposed PLM model over the linear regression model and the conventional backpropagation-based 2LNN model.

The rest of the paper is organized as follows: Section 2 describes in detail the proposed PLM. Section 3 presents the experimental design, and Section 4 presents the empirical results. Section 5 concludes and suggests future work.

## II. PROPOSED PUPIL LEARNING MECHANISM

Without loss of generalization, we consider the regression problem with real-number inputs and use a 2LNN with one output node defined in Eqs. (1) and (2). Table III lists the notation.

$$a_i(\mathbf{x}^c, \mathbf{w}_i^H) \equiv \text{ReLU} \left( w_{i0}^H + \sum_{j=1}^m w_{ij}^H x_j^c \right) \quad (1)$$

$$f(\mathbf{x}^c, \mathbf{w}) \equiv w_0^o + \sum_{i=1}^p w_i^o \text{ReLU} \left( w_{i0}^H + \sum_{j=1}^m w_{ij}^H x_j^c \right) \quad (2)$$

At the  $n$ -th stage, let  $\mathbf{I}(n)$  be the subset of indices of training instances. The learning goal is to find  $\mathbf{w}$  where

$$|e^c| \leq \varepsilon, \forall c \in \mathbf{I}(n).$$

In this study, the  $c$ -th training instance is acceptable if  $|e^c| \leq \varepsilon$ ; otherwise, it is unacceptable. Fig. 1 shows the adaptive GDO

TABLE III  
NOTATION LIST. CHARACTERS IN BOLD REPRESENT COLUMN VECTORS, MATRICES, OR SETS; SUPERSCRIPT  $H$  INDICATES HIDDEN LAYER QUANTITIES; SUPERSCRIPT  $o$  INDICATES OUTPUT LAYER QUANTITIES;  $(\cdot)^T$  DENOTES THE TRANSPOSE OF  $(\cdot)$ .

---

$\text{ReLU}(x) \equiv \max(0, x)$ ;
$m$ : the number of input nodes;
$\mathbf{x}^c \equiv (x_1^c, x_2^c, \dots, x_m^c)^T$ : the $c^{\text{th}}$ input;
$y^c$ : the desired output of the $c^{\text{th}}$ input;
$N$ : the total amount of training instances;
$p$ : the number of adopted hidden nodes;
$w_{i0}^H$ : the bias of $i^{\text{th}}$ hidden nodes;
$w_{ij}^H$ : the weight between the $j^{\text{th}}$ input variable and the $i^{\text{th}}$ hidden nodes;
$\mathbf{w}_i^H \equiv (w_{i0}^H, w_{i1}^H, w_{i2}^H, \dots, w_{im}^H)^T$ ;
$w_0^o$ : the bias value of the output node;
$w_i^o$ : the weight between the $i^{\text{th}}$ hidden node and the output node;
$\mathbf{w}^o \equiv (w_0^o, w_1^o, w_2^o, \dots, w_p^o)^T$ ;
$\mathbf{w}^H \equiv (\mathbf{w}_1^H, \mathbf{w}_2^H, \mathbf{w}_3^H, \dots, \mathbf{w}_p^H)^T$ ;
$\mathbf{w} \equiv \{\mathbf{w}^o, \mathbf{w}^H\}$ ;
$e^c \equiv f(\mathbf{x}^c, \mathbf{w}) - y^c$ .

---

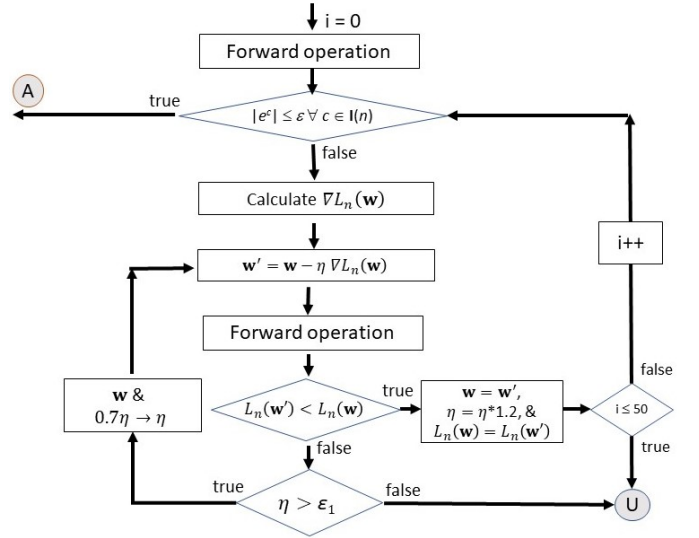


Fig. 1. AGDO implementing an optimizer with an adaptive learning rate  $\eta$ .

(AGDO) implementing an optimizer (e.g., the Adam optimizer [26]) with automatic adjustment of learning rate  $\eta$  to calculate  $\nabla L_n(\mathbf{w})$ , where

$$L_n(\mathbf{w}) \equiv \sum_{c \in \mathbf{I}(n)} (e^c)^2 / n. \quad (3)$$

However, AGDO does not eliminate vanishing gradients and thus may yield an unacceptable 2LNN due to convergence at a local minimum or a saddle point of  $L_n(\mathbf{w})$ . Rate  $\eta$  is smaller than  $\varepsilon_1$  (a tiny value) and the upper iteration bound ( $i \leq 50$ ) is designed to identify unacceptable 2LNNs. In Fig. 1, exit point  $A$  denotes an acceptable 2LNN, and  $U$  denotes an unacceptable 2LNN.

A flowchart for the proposed PLM is shown in Fig. 2.

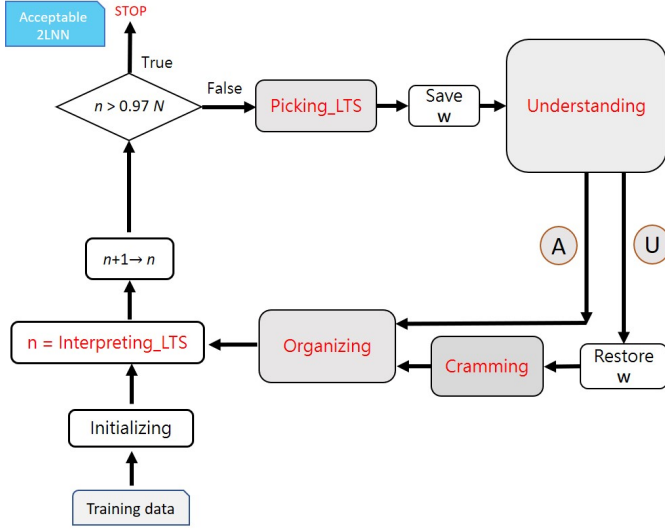


Fig. 2. Flow chart of proposed PLM.

Details are provided as follows. Let

$$D_n = \max_{c \in \mathbf{I}(n)} |f(\mathbf{x}^c, \mathbf{w}) - y^c|,$$

where a 2LNN produces output  $f(\mathbf{x}^c, \mathbf{w})$  when input is  $\mathbf{x}^c$  and the desired output is  $y^c$ .

The initializing module sets up a 2LNN with one hidden node and then uses the GDO to tune the weights and thresholds of the 2LNN with the training instances  $\{(\mathbf{x}^c, y^c) : c \in \mathbf{I}(N)\}$ . Least trimmed squares (LTS) [10], [11] is defined as minimizing  $\sum_{c=1}^q (e^{[c]})^2$ , where only the  $q$  smallest ordered squared residual values are included in the summation, and  $(e^{[c]})^2$  denotes the sorted squared residuals in ascending order  $(e^{[1]})^2 \leq (e^{[2]})^2 \leq \dots \leq (e^{[N]})^2$ . The interpreting\_LTS module implements the LTS principle, which first sorts all training instances by their squared residuals in ascending order as  $(e^{[1]})^2 \leq (e^{[2]})^2 \leq \dots \leq (e^{[N]})^2$ . Second, the interpreting\_LTS module examines the acceptability of every instance (i.e., the  $c^{\text{th}}$  instance is acceptable if  $|e^c| \leq \varepsilon$ ). Third, the interpreting\_LTS module yields the  $n$  value, the number of acceptable instances. Thus,  $D_n \leq \varepsilon$  after the interpreting\_LTS module and before the sequential module ( $n+1 \rightarrow n$ ).

The stopping criterion ( $n < 0.97N$ ) associated with the sequential module indicates that the PLM sequentially learns the training instances until more than  $\lfloor 0.97N \rfloor$  instances are acceptable, where  $\lfloor x \rfloor$  is the largest integer less than or equal to  $x$ . In other words, the PLM learns merely the majority of training data (with at most  $\lfloor 0.97N \rfloor$  data), but the learning terminates when the resultant 2LNN renders more than  $\lfloor 0.97N \rfloor$  (training) data acceptable.

For  $n < 0.97N$ , the picking\_LTS module selects the first  $n$  instances with the smallest squared residuals as the training instances, for which  $\mathbf{I}(n)$  is the set of instance indices. Note that the  $\mathbf{I}(n)$  subset contains merely one unacceptable instance: the  $[n]$ -th instance. This prevents the PLM not only from learning instances with much larger squared residuals that are difficult for the learning mechanism to understand, but also from causing more cramming occurrences, which may

TABLE IV  
CRAMMING MODULE

Step 1: Select a tiny positive number  $\zeta$  and randomly generate an  $m$ -vector  $\gamma$  of length one such that

$$\gamma^T (\mathbf{x}^c - \mathbf{x}^{[n]}) \neq 0,$$

$$(\zeta + \gamma^T (\mathbf{x}^c - \mathbf{x}^{[n]})) (\zeta - \gamma^T (\mathbf{x}^c - \mathbf{x}^{[n]})) < 0, \\ \forall c \in \mathbf{I}(n) \setminus \{[n]\}.$$

Step 2: Let  $p+3 \rightarrow p$ , add three new hidden nodes  $p-2$ ,  $p-1$ , and  $p$  to the current 2LNN, and assign their associated weights and thresholds as follows to ensure that  $|e^c| \leq \varepsilon, \forall c \in \mathbf{I}(n)$  is true:

$$\begin{aligned} \mathbf{w}_{p-2}^H &= \gamma, w_{p-2,0}^H = \zeta - \gamma^T \mathbf{x}^{[n]}, \\ w_{p-2}^o &= \frac{y^{[n]} - w_0^o - \sum_{i=1}^{p-3} w_i^o a_i^{[n]}}{\zeta}, \\ \mathbf{w}_{p-1}^H &= \gamma, w_{p-1,0}^H = -\gamma^T \mathbf{x}^{[n]}, \\ w_{p-1}^o &= \frac{-2(y^{[n]} - w_0^o - \sum_{i=1}^{p-3} w_i^o a_i^{[n]})}{\zeta}, \\ \mathbf{w}_p^H &= \gamma, w_{p,0}^H = -\zeta - \gamma^T \mathbf{x}^{[n]}, \\ w_p^o &= \frac{y^{[n]} - w_0^o - \sum_{i=1}^{p-3} w_i^o a_i^{[n]}}{\zeta}. \end{aligned}$$

lead to too many hidden nodes and thus overfitting. Then,  $\mathbf{w}$  is saved such that when  $\mathbf{w}$  is restored we can resume the 2LNN estimate to the scenario in which there is only one unacceptable instance in the  $\mathbf{I}(n)$  subset. The interpreting\_LTS module and picking\_LTS modules together yield a learning sequence in which easy instances are learned first.

The understanding module implements the AGDO of Fig. 1 with the loss function defined at equation 3 to learn all training instances of  $\mathbf{I}(n)$ . If the understanding module results in an acceptable 2LNN estimate, then  $D_n \leq \varepsilon$ . Otherwise, the PLM restores  $\mathbf{w}$  and triggers the cramming module of Table IV to process instance  $[n]$  by adding three hidden nodes so the 2LNN now has  $p$  hidden nodes. That is, with the restore module,

$$\begin{aligned} \left( y^{[n]} - w_0^o - \sum_{i=1}^{p-3} w_i^o a_i^{[n]} \right)^2 &> \varepsilon^2, \\ \left( y^c - w_0^o - \sum_{i=1}^{p-3} w_i^o a_i^c \right)^2 &\leq \varepsilon^2, \forall c \in \mathbf{I}(n) \setminus \{[n]\}. \end{aligned}$$

To compute  $D_n$ , we first consider instance  $[n]$  as input and show that the output of 2LNN,  $f(\mathbf{x}^{[n]}, \mathbf{w})$ , equals exactly the desired output  $y^{[n]}$ . By Step 2 of the cramming module,

$$\begin{aligned} a_{p-2}^{[n]} &= \text{ReLU} \left( w_{p-2,0}^H + \sum_{j=1}^m w_{p-2,j}^H x_j^{[n]} \right) \\ &= \text{ReLU} \left( \zeta + \gamma^T (\mathbf{x}^{[n]} - \mathbf{x}^{[n]}) \right) \\ &= \zeta. \end{aligned}$$

Similarly,

$$a_{p-1}^{[n]} = \text{ReLU} \left( \gamma^T \left( \mathbf{x}^{[n]} - \mathbf{x}^{[n]} \right) \right) = 0,$$

$$a_p^{[n]} = \text{ReLU} \left( -\zeta + \gamma^T \left( \mathbf{x}^{[n]} - \mathbf{x}^{[n]} \right) \right) = 0.$$

Since  $f(\mathbf{x}^{[n]}, \mathbf{w})$  equals the contribution of the first  $p-3$  hidden nodes to the output node plus contribution of the  $p-2^{th}$ ,  $p-1^{th}$  and  $p^{th}$  hidden nodes to the output node, by the assignment of  $w_{p-2}^o$ ,  $w_{p-1}^o$ , and  $w_p^o$  in Step 2 of the cramming module,

$$\begin{aligned} f(\mathbf{x}^{[n]}, \mathbf{w}) &= w_0^o + \sum_{i=1}^{p-3} w_i^o a_i^{[n]} \\ &\quad + \zeta \left( \frac{y^{[n]} - w_0^o - \sum_{i=1}^{p-3} w_i^o a_i^{[n]}}{\zeta} \right) \\ &= y^{[n]} \end{aligned}$$

Thus, when instance  $[n]$  is input and the cramming module is executed, no additional effect on  $D_n$  is introduced by instance  $[n]$ .

With regard to the effect on  $D_n$  of instances  $c \in \mathbf{I}(n) \setminus \{[n]\}$ , we argue that these instances will not increase  $D_n$ . The change in the output node is due to the inputs from the  $p-2^{th}$ ,  $p-1^{th}$  and  $p^{th}$  hidden nodes. If instance  $c$  is input,

$$\begin{aligned} a_{p-2}^c &= \text{ReLU} \left( w_{p-2,0}^H + \sum_{j=1}^m w_{p-2,j}^H x_j^c \right) \\ &= \text{ReLU} \left( \zeta + \gamma^T \left( \mathbf{x}^c - \mathbf{x}^{[n]} \right) \right). \end{aligned}$$

Similarly,

$$a_{p-1}^c = \text{ReLU} \left( \gamma^T \left( \mathbf{x}^c - \mathbf{x}^{[n]} \right) \right),$$

$$a_p^c = \text{ReLU} \left( -\zeta + \gamma^T \left( \mathbf{x}^c - \mathbf{x}^{[n]} \right) \right).$$

By the assignment of  $w_{p-2}^o$ ,  $w_{p-1}^o$ , and  $w_p^o$  in Step 2 of the cramming module,

$$\begin{aligned} \sum_{i=p-2}^p w_i^o a_i^c &= w_p^o \left[ \text{ReLU} \left( \zeta + \gamma^T \left( \mathbf{x}^c - \mathbf{x}^{[n]} \right) \right) \right. \\ &\quad \left. - 2\text{ReLU} \left( \gamma^T \left( \mathbf{x}^c - \mathbf{x}^{[n]} \right) \right) \right. \\ &\quad \left. + \text{ReLU} \left( -\zeta + \gamma^T \left( \mathbf{x}^c - \mathbf{x}^{[n]} \right) \right) \right] \\ &= g(c). \end{aligned}$$

In Step 1 of the Table IV, we have set  $\zeta$  and  $\gamma$ ; therefore,  $g(c) = 0, \forall c \in \mathbf{I}(n) \setminus \{[n]\}$ . Thus, the cramming module makes the change in the output node in all instances  $c \in \mathbf{I}(n) \setminus \{[n]\}$  equal to zero. In other words,  $D_n \leq \varepsilon$ , and by adding three extra hidden nodes with the ReLU activation function, the cramming module renders  $|e^c| \leq \varepsilon, \forall c \in \mathbf{I}(n)$  true immediately.

The organizing module of Fig. 3 reduces overfitting by identifying and pruning irrelevant hidden nodes [9]. As shown in Fig. 3, the organizing module first implements the regularizing

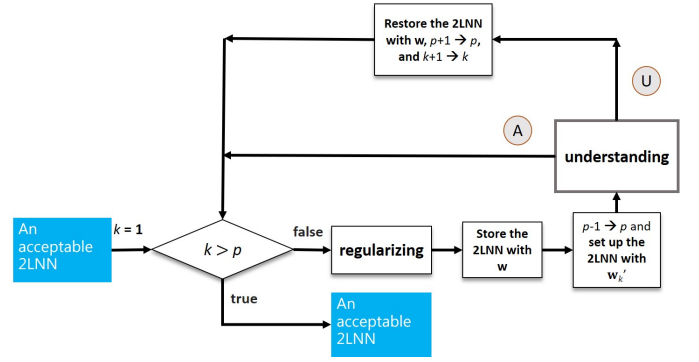


Fig. 3. Organizing module where  $\mathbf{w}'_k \equiv \mathbf{w} \setminus \{w_k^0, \mathbf{w}_k^H\}$ .

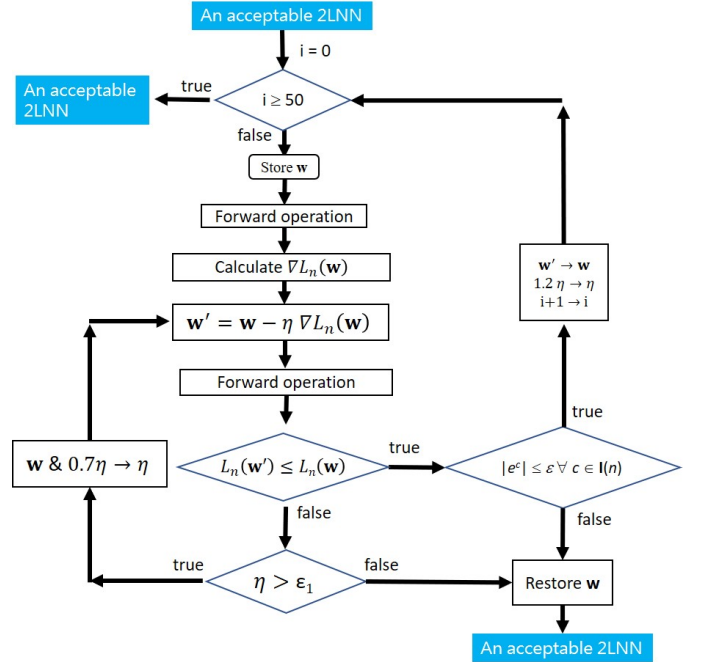


Fig. 4. Regularizing module.

module of Fig. 4, which regularizes the weights and thresholds by means of the loss function

$$Lr_n(\mathbf{w}) \equiv \sum_{c \in \mathbf{I}(n)} (e^c)^2 / n + \lambda \|\mathbf{w}\|^2, \quad (4)$$

where  $\lambda$  is the regularization coefficient. In contrast with the AGDO of Fig. 1, the regularizing module deploys AGDO to tune  $\mathbf{w}$  while keeping  $|e^c| \leq \varepsilon, \forall c \in \mathbf{I}(n)$  true. Namely, with the loss function defined at equation 4, the regularizing module attempts to prevent large-magnitude weights and thresholds while keeping  $|e^c| \leq \varepsilon, \forall c \in \mathbf{I}(n)$  true. Thus  $D_n \leq \varepsilon$  after implementing the regularizing module.

After implementing the regularizing module, the organizing module stores  $\mathbf{w}$  and creates a new 2LNN with  $p-1 \rightarrow p$ , the  $k^{th}$  hidden node is removed, and the weights and thresholds are assigned as  $\mathbf{w}'_k \equiv \mathbf{w} \setminus \{w_k^0, \mathbf{w}_k^H\}$ . Then, the understanding module with the AGDO of Fig. 1 is applied to this new 2LNN to  $\min_{\mathbf{w}'_k} (L_n(\mathbf{w}'_k))$  and  $\mathbf{w}'_k$  is adjusted with the loss

function at equation 3. If the understanding module yields an acceptable 2LNN, the afore-mentioned process is applied again to the next hidden node. If the understanding module yields an unacceptable 2LNN, then the stored 2LNN with  $w$  is restored,  $p + 1 \rightarrow p$ ,  $k + 1 \rightarrow k$ , and the afore-mentioned process is again applied to the next hidden node.

The organizing module ensures that  $|e^c| \leq \varepsilon, \forall c \in \mathbf{I}(n)$  is true. Thus  $D_n \leq \varepsilon$  and we have the following Lemma. In addition, the organizing module helps to identify and prune irrelevant hidden nodes to reduce overfitting in the resulting 2LNN.

**Lemma:** For all  $n$ , the PLM yields a 2LNN with  $D_n \leq \varepsilon$ .

Note that when learning a new (unacquainted) instance, the PLM results in an acceptable 2LNN via one of the following two routes:

- 1) Understanding route by implementing the AGDO of Fig. 1 to yield an acceptable 2LNN.
- 2) Cramming route by implementing the cramming module of Table IV to yield an acceptable 2LNN.

### III. EXPERIMENT DESIGN

To validate the proposed PLM, we conducted an experiment on copper price forecasting to determine the following: 1) whether the understanding and cramming routes are effective; 2) whether the sequential module and the cramming module effectively address the vanishing gradient problem; and 3) whether LTS and the organizing arrangements reduce overfitting. This study further examines whether the total PLM training time is reasonable and whether the PLM yields better forecast accuracy than other tools mentioned in the literature. Based upon the literature review regarding copper price forecasting, this study identifies the 18 input variables and the four-week-ahead-forecast output variables shown in Table V. Therefore,  $m = 18$ .

The dataset includes 471 weekly copper prices of Yangtze River (YR) nonferrous metals from 2011/10/31 to 2020/12/21. The average (AVG) and standard deviation (SD) of  $y$  are 48,358.75 RMB/ton and 6,183.97 RMB/ton, respectively. In this study,  $y$  is divided by 100,000 and its values fall between 0 and 1. Twenty datasets were generated by randomly picking 282 instances, which comprised approximately 60% of the dataset of 471 instances, as the training data and the remaining 189 instances were testing data. Therefore,  $N = 282$  and the learned majority of training data was at most 272 data.

To explore the study objectives, we used four different versions of the PLM shown in Table VI.  $LTS_n^N$  indicates that both the interpreting module and the picking module follow the LTS principle to yield the  $n$  value, which is set to the count of acceptable instances, and pick the first  $n$  instances, respectively, whereas  $PO_n^N$  denotes that both the interpreting module and the picking module follow the pre-ordered principle. Organizing( $x$ ) indicates that the organizing module regularizes weights for at most  $x$  epochs. These versions were implemented with the PyTorch framework and GPU hardware shown in Table VII.

The  $\varepsilon$  value for the interpreting, understanding, and organizing modules is set to 0.04836 (i.e., 10% of the average

TABLE V  
INPUT AND OUTPUT VARIABLES.

Variable	Description	Ref.
$x_1^t$	Weekly crude oil price of New York Mercantile Exchange at time epoch $t$	[27], [28]
$x_2^t$	Weekly copper spot price of YR nonferrous metals at epoch $t$	
$x_3^t$	Weekly copper spot price of YR nonferrous metals at epoch $t - 1$	
$x_4^t$	Weekly copper spot price of YR nonferrous metals at epoch $t - 2$	
$x_5^t$	Weekly copper spot price of YR nonferrous metals at epoch $t - 3$	
$x_6^t$	Weekly copper spot price of London Metal Exchange at epoch $t$	[27]
$x_7^t$	Weekly gold spot price of FX Broker at epoch $t$	[27]
$x_8^t$	Weekly silver spot price of FX Broker at epoch $t$	[27]
$x_9^t$	Weekly nickel spot price of London Metal Exchange at epoch $t$	[27]
$x_{10}^t$	Weekly aluminum spot price of London Metal Exchange at epoch $t$	[27]
$x_{11}^t$	Weekly zinc spot price of London Metal Exchange at epoch $t$	[27]
$x_{12}^t$	Weekly iron spot price of London Metal Exchange at epoch $t$	[27]
$x_{13}^t$	US inflation rates at epoch $t$	[27], [29]
$x_{14}^t$	China inflation rates at epoch $t$	[27], [29]
$x_{15}^t$	Weekly USD/CLP dollar exchange rate at epoch $t$	[27], [30]
$x_{16}^t$	Weekly USD/PEN dollar exchange rate at epoch $t$	[27], [30]
$x_{17}^t$	Weekly USD/RMB dollar exchange rate at epoch $t$	[27], [30]
$x_{18}^t$	Weekly USD/EURO dollar exchange rate at epoch $t$	[27], [30]
$y^t$	Weekly copper spot price of YR nonferrous metals at epoch $t + 4$	[31]

TABLE VI  
PLM VERSIONS AND SETTINGS.

Version	Interpreting module and picking module	Organizing module
PLM-PO-100	$PO_n^N$	organizing(100)
PLM-LTS-0	$LTS_n^N$	organizing(0)
PLM-LTS-100	$LTS_n^N$	organizing(100)
PLM-LTS-500	$LTS_n^N$	organizing(500)

TABLE VII  
COMPUTATIONAL ENVIRONMENT.

OS	Ubuntu 20.04.4 LTS
Programming Language	Python 3.7.7
Pytorch version	1.11.0
IDE	PyCharm, Jupyter Notebook
GPU	GeForce RTX 2080
RAM	DDR-4 8G*4

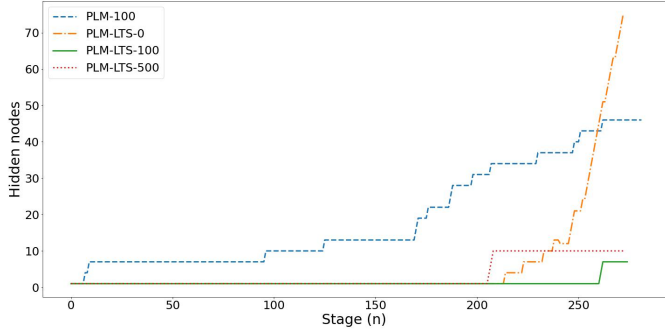


Fig. 5. Number of hidden nodes during learning process of  $10^{th}$  training dataset.

TABLE VIII  
OCCURRENCE FREQUENCIES WITH UNDERSTANDING ROUTE.

	PLM-PO-100	PLM-LTS-0	PLM-LTS-100	PLM-LTS-500
Min	30.41%	10.53%	27.78%	14.29%
Max	77.14%	77.42%	78.57%	100.00%
Avg	58.26%	44.31%	56.91%	57.32%
SD	12.38%	19.67%	13.73%	22.01%

of the normalized  $y$ ). Thus, when  $-0.04836 \leq f(\mathbf{x}^c, \mathbf{w}) - y^c \leq 0.04836$ , the  $c^{th}$  instance is classified as acceptable; otherwise, it is unacceptable. The initial learning rates  $\eta$  in the understanding and organizing modules are set to  $1e-2$  and  $1e-3$ , respectively. Both understanding and organizing modules have an  $\varepsilon_1$  value of  $1e-7$ .

#### IV. EXPERIMENTAL RESULTS

Fig. 5 shows the evolution of the number of hidden nodes during the learning process of the  $10^{th}$  training dataset for the four versions. As expected, cramming was triggered several times in the early stage of the learning process of PLM-PO-100. In contrast, in the early stage of the learning process of PLM-LTS-100 (and other PLM-LTS-xxx versions), the cramming module was not often triggered. Furthermore, compared with the learning process of PLM-LTS-0 shown in Fig. 5 with those of PLM-LTS-100 and PLM-LTS-500, the non-zero regularizing epochs reduce the frequency of cramming.

Tables VIII and IX show the occurrence frequencies of the understanding route and cramming route over twenty training datasets for the four versions. All non-zero minimal occurrence frequencies of the cramming route for PLM-PO-100, PLM-LTS-0, and PLM-LTS-100 shown in Table IX indicate an unavoidable vanishing gradient problem for the understanding module. The cramming module is indeed required to address the vanishing gradient. It is worth noting that a zero-value minimal occurrence frequency of the cramming route for PLM-LTS-500 is shown in Table IX.

Table X shows the total number of hidden nodes at the end of learning over twenty training datasets for the four versions. Comparing the PLM-PO-100 and PLM-LTS-100 results, we see that LTS indeed reduces the AVG and SD of the number of hidden nodes and thus reduces overfitting. Comparing the PLM-LTS results, we also see that the more regularizing

TABLE IX  
OCCURRENCE FREQUENCIES WITH CRAMMING ROUTE.

	PLM-PO-100	PLM-LTS-0	PLM-LTS-100	PLM-LTS-500
Min	22.86%	22.58%	21.43%	0.00%
Max	69.59%	89.47%	72.22%	85.71%
Avg	41.74%	55.70%	43.09%	42.68%
SD	12.38%	19.67%	13.73%	22.01%

TABLE X  
HIDDEN NODES AT THE END OF LEARNING.

	PLM-PO-100	PLM-LTS-0	PLM-LTS-100	PLM-LTS-500
Min	16	13	4	1
Max	301	96	66	55
Avg	52.15	44.00	29.65	22.05
SD	60.93	26.92	19.24	15.83

epochs, the better the minimum, maximum, AVG, and SD of the number of hidden nodes. In other words, regularizing epochs reduce overfitting.

Table XI shows the number of hidden nodes pruned during the learning process over twenty training datasets for the four versions. It seems that for these four versions, the organizing module indeed works frequently, but does not guarantee successful pruning over a single learning process since the minimal number of hidden nodes pruned within the learning process over twenty training datasets are all zero. In terms of the average number of hidden nodes pruned over the learning process, PLM-LTS-500 has the fewest and PLM-PO-100 has the most.

Table XII shows the training times over twenty training datasets for the four versions. PLM-LTS-500 has the smallest average and standard deviation, and PLM-PO-100 has the largest. Taken together with the results of Tables X and XII, we observe that the adoption of LTS and the “longer” regularizing module reduce the average and standard deviation of the hidden nodes and thus the training time.

Table XIII shows the mean absolute error (MAE) over twenty datasets for the four versions. Here, the MAE of the

TABLE XI  
HIDDEN NODES PRUNED OVER LEARNING PROCESS.

	PLM-PO-100	PLM-LTS-0	PLM-LTS-100	PLM-LTS-500
Min	0	0	0	0
Max	9	4	4	4
Avg	2.10	0.95	0.90	0.79
SD	2.27	1.32	1.33	1.38

TABLE XII  
TRAINING TIME (IN SECOND).

	PLM-PO-100	PLM-LTS-0	PLM-LTS-100	PLM-LTS-500
Min	51.58	50.81	13.75	12.50
Max	8619.52	1753.65	1126.95	832.77
Avg	766.15	589.21	430.30	222.55
SD	1862.51	528.63	377.19	247.92

TABLE XIII  
MAE RESULTS.

	PLM-PO-100	PLM-LTS-0	PLM-LTS-100	PLM-LTS-500
Training Data (majority)				
Min	0.0097	0.0081	0.0097	0.0092
Max	0.0132	0.0156	0.0166	0.0148
Avg	0.0116	0.0119	0.0129	0.0132
SD	0.0011	0.0023	0.0016	0.0014
Training Data (non-majority)				
Min	0.0399	0.0572	0.0476	0.0501
Max	0.1379	0.1883	0.1306	0.1389
Avg	0.0501	0.0951	0.0776	0.0739
SD	0.0212	0.0352	0.0211	0.0234
Training Data (Testing data)				
Min	0.0155	0.0153	0.0152	0.0144
Max	0.0783	0.0332	0.0225	0.0190
Avg	0.0217	0.0201	0.0185	0.0170
SD	0.0135	0.0043	0.0021	0.0014
Ratio of the average MAE of testing data to that of training data (majority)				
	1.870	1.689	1.434	1.288

training data (majority) is calculated for the first 272 training data sorted by the LTS principle, the MAE of the training data (non-majority) is calculated for the remaining 10 training data, and the MAE of the testing data is calculated for the 189 testing data. The last row of Table XIII displays the ratios of the average MAE of the testing data to the average MAE of the training data (majority), suggesting that the smaller the ratio, the less overfitting there is. Table XIII shows that PLM-PO-100 has the highest ratio and PLM-LTS-500 has the smallest ratio. LTS and the “longer” regularizing module appear to reduce overfitting.

The empirical results shown in Tables VIII to XIII indicate that, in terms of reducing the vanishing gradient and overfitting, the best version is PLM-LTS-500. We further compared the predictive performance of PLM-LTS-500 with two popular modeling tools: the linear regression model and the conventional backpropagation-based 2LNN model.

The experiment was conducted as follows. First, the linear regression model was used to learn 20 datasets, each of which had 282 training data and 189 testing data. The obtained average MAE for the training and testing data were 0.0142 and 0.0140, respectively. Thus, we set the  $\varepsilon$  value of the learning goal of both the conventional 2LNN model and the PLM-LTS-500 model to 0.0282 (i.e., approximately equal to  $2 \times$  average MAE of the linear regression model). Then, the PLM-LTS-500 model was used to learn the 20 datasets. The final number of hidden nodes for each dataset was recorded and used to set the (fixed) number of hidden nodes for the conventional 2LNN model for each dataset, which is termed the 2LNN\_v model. The (fixed) number of hidden nodes for the other two conventional 2LNN models were set to 13 and 23; the initial weights were set randomly. The number of learning epochs of the backpropagation learning algorithm used to tune the weights of the three conventional 2LNN models was 500.

Table XIV shows the MAE results over twenty datasets for

TABLE XIV  
MAE RESULTS FOR LINEAR REGRESSION MODEL, 2LNN\_13, 2LNN\_23, 2LNN\_v AND PLM-LTS-500.

	Linear Regression	2LNN_13	2LNN_23	2LNN_v	PLM-LTS-500
Training Data					
Min	0.0134	0.0073	0.0069	0.0073	0.0066
Max	0.0150	0.2056	0.2321	0.2044	0.0090
Avg	0.0142	0.0820	0.0741	0.0974	0.0079
SD	0.0005	0.0645	0.0667	0.0517	0.0008
Testing Data					
Min	0.0127	0.0114	0.0107	0.0123	0.0121
Max	0.0155	0.1855	0.2172	0.2181	0.0149
Avg	0.0140	0.0813	0.0744	0.0980	0.0138
SD	0.0008	0.0569	0.0620	0.0509	0.0008

the linear regression model, the conventional 2LNN model with 13 hidden nodes (2LNN\_13), the conventional 2LNN model with 23 hidden nodes (2LNN\_23), the 2LNN\_v model, and the PLM-LTS-500 model. Here, for the PLM-LTS-500 model, the training MAE was calculated for the first 272 training data sorted by LTS and the test MAE was calculated for all 189 testing data. Table XII shows that all MAE results for PLM-LT-500 were better than those for the conventional 2LNN models. Thus, PLM yields much better learning performance than backpropagation learning. Furthermore, all MAE results for PLM-LT-500 are better than those for the linear regression model under the condition that the  $\varepsilon$  value of the learning goal is approximately equal to twice the average MAE of the linear regression model. Thus, the proposed PLM-LTS-500 model yields performance superior to the linear regression model and the conventional 2LNN model.

## V. CONCLUSION

In this study we conduct experiments using a copper price forecasting dataset to validate and evaluate the proposed PLM. The experimental results shown in Fig. 5 validate that the proposed PLM indeed implements the pupil learning procedure of Table I with interpreting, picking, understanding, cramming, and organizing. For instance, the interpreting\_LTS module separates all instances into the acquainted instances and the unacquainted instances. The picking\_LTS module picks unacquainted instances one by one such that easy (unacquainted) instances are learned first and similar instances are grouped together. The cramming module enforces memorization when the understanding module cannot learn a picked unacquainted instance. The organizing module comprehends all learned instances for a concise knowledge system. The empirical results show that these modules constitute sequential learning, adaptive learning, perfect learning, and less-overfitted learning. The empirical results also verify the effectiveness of the proposed PLM in handling both vanishing gradients and overfitting.

The empirical results of the second experiment also attest the superiority of the PLM model over the linear regression model and the conventional backpropagation-based 2LNN model. The empirical results indicate that the learning process

resulting from the PLM varies based on the hyperparameter arrangements of all modules. One future work is to identify the best hyperparameters for all modules for specific applications when the PLM is adopted.

## VI. ACKNOWLEDGMENTS

This research was supported by the Ministry of Science and Technology, Taiwan, under grants MOST 109-2410-H-004-067-MY2 and 111-2410-H-004-063-MY3.

## REFERENCES

- [1] D. Rumelhart, G. Hinton, and R. Williams, "Learning internal representations by error propagation," *readings in cognitive science: A perspective from psychology and artificial intelligence*, 2013.
- [2] B. Pérez-Sánchez, O. Fontenla-Romero, and B. Guijarro-Berdiñas, "A review of adaptive online learning for artificial neural networks," *Artificial Intelligence Review*, vol. 49, pp. 281–299, 2018.
- [3] L. Ma and K. Khorasani, "A new strategy for adaptively constructing multilayer feedforward neural networks," *Neurocomputing*, vol. 51, pp. 361–385, 2003.
- [4] M. Mezard and J.-P. Nadal, "Learning in feedforward layered networks: The tiling algorithm," *Journal of Physics A: Mathematical and General*, vol. 22, no. 12, p. 2191, 1989.
- [5] T. Kusuma and M. M. Brown, "Cascade-correlation learning architecture for first-break picking and automated trace editing," in *SEG Technical Program Expanded Abstracts 1992*. Society of Exploration Geophysicists, 1992, pp. 1136–1139.
- [6] Y. Q. Chen, D. W. Thomas, and M. S. Nixon, "Generating-shrinking algorithm for learning arbitrary classification," *Neural Networks*, vol. 7, no. 9, pp. 1477–1489, 1994.
- [7] M. Frean, "The upstart algorithm: A method for constructing and training feedforward neural networks," *Neural computation*, vol. 2, no. 2, pp. 198–209, 1990.
- [8] R. Tsaih, "The softening learning procedure," *Mathematical and computer modelling*, vol. 18, no. 8, pp. 61–64, 1993.
- [9] —, "An explanation of reasoning neural networks," *Mathematical and Computer Modelling*, vol. 28, no. 2, pp. 37–44, 1998.
- [10] R.-H. Tsaih and T.-C. Cheng, "A resistant learning procedure for coping with outliers," *Annals of Mathematics and Artificial Intelligence*, vol. 57, pp. 161–180, 2009.
- [11] Y.-H. Tsai, Y.-J. Jheng, and R.-H. Tsaih, "The cramming, softening and integrating learning algorithm with parametric relu activation function for binary input/output problems," in *2019 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2019, pp. 1–7.
- [12] T. Dietterich, "Overfitting and undercomputing in machine learning," *ACM computing surveys (CSUR)*, vol. 27, no. 3, pp. 326–327, 1995.
- [13] D. M. Hawkins, "The problem of overfitting," *Journal of chemical information and computer sciences*, vol. 44, no. 1, pp. 1–12, 2004.
- [14] S.-i. Amari, "Backpropagation and stochastic gradient descent method," *Neurocomputing*, vol. 5, no. 4-5, pp. 185–196, 1993.
- [15] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin *et al.*, "Tensorflow: Large-scale machine learning on heterogeneous distributed systems," *arXiv preprint arXiv:1603.04467*, 2016.
- [16] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard *et al.*, "Tensorflow: A system for large-scale machine learning. 12th {USENIX} symposium on operating systems design and implementation ({OSDI} 16), 265–283," *Google Scholar Google Scholar Digital Library Digital Library*, 2016.
- [17] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga *et al.*, "Pytorch: An imperative style, high-performance deep learning library," *Advances in neural information processing systems*, vol. 32, 2019.
- [18] Y. Bengio, A. Courville, and P. Vincent, "Representation learning: A review and new perspectives," *IEEE transactions on pattern analysis and machine intelligence*, vol. 35, no. 8, pp. 1798–1828, 2013.
- [19] Y. LeCun, Y. Bengio, G. Hinton *et al.*, "Deep learning. nature, 521 (7553), 436–444," *Google Scholar Google Scholar Cross Ref Cross Ref*, p. 25, 2015.
- [20] S.-M. Lu, D.-P. Li, and Y.-J. Liu, "Adaptive neural network control for uncertain time-varying state constrained robotics systems," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 49, no. 12, pp. 2511–2518, 2017.
- [21] L. Ma and L. Liu, "Adaptive neural network control design for uncertain nonstrict feedback nonlinear system with state constraints," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 51, no. 6, pp. 3678–3686, 2019.
- [22] B. Niu, H. Li, Z. Zhang, J. Li, T. Hayat, and F. E. Alsaadi, "Adaptive neural-network-based dynamic surface control for stochastic interconnected nonlinear nonstrict-feedback systems with dead zone," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 49, no. 7, pp. 1386–1398, 2018.
- [23] X. Wang, A. Bao, E. Lv, and Y. Cheng, "Multiscale multipath ensemble convolutional neural network," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 51, no. 9, pp. 5918–5928, 2020.
- [24] K. Hara, D. Saito, and H. Shouno, "Analysis of function of rectified linear unit used in deep learning," in *2015 international joint conference on neural networks (IJCNN)*. IEEE, 2015, pp. 1–8.
- [25] B. Xu, N. Wang, T. Chen, and M. Li, "Empirical evaluation of rectified activations in convolutional network," *arXiv preprint arXiv:1505.00853*, 2015.
- [26] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.
- [27] Z. Alameer, M. A. Elaziz, A. A. Ewees, H. Ye, and Z. Jianhua, "Forecasting copper prices using hybrid adaptive neuro-fuzzy inference system and genetic algorithms," *Natural Resources Research*, vol. 28, pp. 1385–1401, 2019.
- [28] C. Liu, Z. Hu, Y. Li, and S. Liu, "Forecasting copper prices by decision tree learning," *Resources Policy*, vol. 52, pp. 427–434, 2017.
- [29] L. T. Orlowski, "Volatility of commodity futures prices and market-implied inflation expectations," *Journal of International Financial Markets, Institutions and Money*, vol. 51, pp. 133–141, 2017.
- [30] T. Wang and C. Wang, "The spillover effects of china's industrial growth on price changes of base metal," *Resources Policy*, vol. 61, pp. 375–384, 2019.
- [31] G. Astudillo, R. Carrasco, C. Fernández-Campusano, and M. Chacón, "Copper price prediction using support vector regression technique," *applied sciences*, vol. 10, no. 19, p. 6648, 2020.

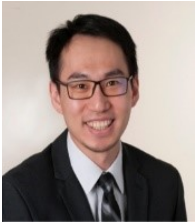


**Rua-Huan Tsaih** received his PhD in operations research in 1991 from the University of California, Berkeley. His research interests include neural networks, decision support systems, operations research, and service innovation. His recent work has been published in *Decision Support Systems*, *Computer Communications*, *Mathematical and Computer Modelling*, *Computer & Operation Research*, *Communications of the ACM*, *IT Professional*, *Expert Systems with Applications*, *Industrial Management and Data Systems*, *Malaysian Journal of Library &*

*Information Science*, *NTU Management Review*, and *Review of Securities and Futures Markets*.



**Yu-Hang Chien** is an undergraduate student at National Chengchi University of Taipei, Taiwan. His research interests include neural networks, computer vision, and distributed systems.



**Shih-Yi Chien** is an Associate Professor in the Department of Management Information Systems at National Chengchi University, Taiwan. His current research interests include human-robot interaction, human-automation collaboration, trust in automation, and XAI. His research has appeared in *IEEE Transactions on Human-Machine Systems*, *ACM Transactions on Interactive Intelligent Systems*, *International Journal of Human-Computer Studies*, *International Journal of Human-Computer Interaction*, *Electronic Commerce Research and Applications*,

*Journal of Cognitive Engineering and Decision Making*, *Journal of Intelligent and Robotic Systems*, *Human Factors*, and others.