Composite Neural Network: Theory and Application to PM2.5 Prediction

Ming-Chuan Yang and Meng Chang Chen

Institute of Information Science, Academia Sinica, Taiwan

{mingchuan,mcc}@iis.sinica.edu.tw

Abstract

This work investigates the framework and statistical performance guarantee of the composite neural network, which is composed of a collection of pre-trained and non-instantiated neural network models connected as a rooted directed acyclic graph, for solving complicated applications. A pre-trained neural network model is generally well trained, targeted to approximate a specific function. The advantages of adopting a pre-trained model as a component in composing a complicated neural network are two-fold. One is benefiting from the intelligence and diligence of domain experts, and the other is saving effort in data acquisition as well as computing resources and time for model training. Despite a general belief that a composite neural network may perform better than any a single component, the overall performance characteristics are not clear. In this work, we propose the framework of a composite network, and prove that a composite neural network performs better than any of its pre-trained components with a high probability.

In the study, we explore a complicated application—PM2.5 prediction—to support the correctness of the proposed composite network theory. In the empirical evaluations of PM2.5 prediction, the constructed composite neural network models perform better than other machine learning models.

Index Terms

deep learning, pre-trained component, composite neural network, PM2.5 prediction.

I. Introduction

Deep learning has seen great success in dealing with natural signals such as images and voices as well as artificial signals such as natural language, whereas it is still in the early stages of handling complicated social and natural applications shaped by diverse factors (e.g., stock market prediction [1]) or that result from complicated natural processes (e.g., PM2.5 pollution level prediction [2]). Common to these complicated applications is their unbounded applicable data sources, which may not be available all at once, and their processes, which are difficult to learn from limited data. Consequently, their neural network based solutions often require frequent revisions as more relevant data are available or more data is made available, or the understanding of the process is enhanced. Although neural networks can approximate arbitrary functions [3], competent neural networks for complicated applications are unrealistic for the above reasons, which motivates this study to devise an effective, realistic approach for such applications.

The obvious drawbacks of traditional approaches to suitable neural network models include a lack of flexibility given new data sources and knowledge, difficulty in improving problem modeling and decomposition, and an inability to employ the proven efforts of others. The main idea of the proposed composite neural network is to compose several neural network models, especially pre-trained models (i.e., neural network models with instantiated weights), based on the availability of data and domain knowledge, to solve complicated applications.

An emerging trend in deep learning solution development is to employ well-crafted pre-trained neural networks, especially for use as a specific function/component to synthesize a neural network model. Many popular pre-trained neural network models are fine-tuned with adequate training data and made available to the public either as open-source or commercial products. In practice, training a large neural network is infeasible due to the limitations of computing resources. Pre-trained components may alleviate the problem by decomposing the problem into several sub-problems, each of which can be solved by a neural network component which can be trained separately. The advantages of adopting a pre-trained model in composing a complicated neural network are two-fold. One is benefiting from the intelligence and diligence of domain experts, and the other is saving effort in data acquisition as well as computing resources and time for model training.

During the training phase of a composite network, the weights of pre-trained models are frozen to maintain their original quality, and to save training time for less trainable parameters, whereas the weights of their incoming and outgoing edges are trainable. Note that a user may choose the weights of a pre-trained component trainable for their particular purpose. For instance, in transfer learning, the weights of the pre-trained network may be used as initial values in the training phase of the overall neural network. Ensemble learning [4] and transfer learning [5] both apply additional data and neural network models to improve accuracy. In deep learning, ensemble learning (Fig. 1(a)) employs multiple neural networks together to make decisions whereas transfer learning (Fig. 1(b)) applies knowledge learned from other neural networks to assist in solving the
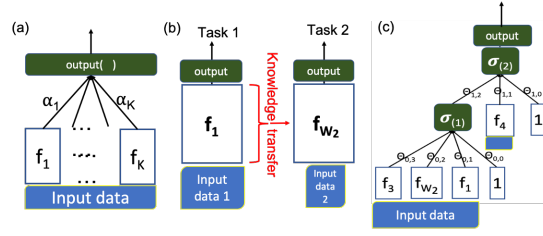
Fig. 1. Illustrations of (a) ensemble learning, (b) transfer learning, and (c) composite neural network.
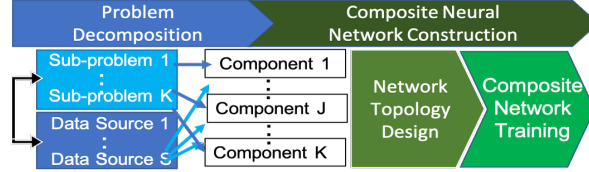

Fig. 2. Framework of Composite Neural Network Construction

original problem. Although all the models consider pre-trained components, the ensemble learning basically adopts homogeneity learners (i.e., learners for the same function), and transfer learning methods are applied in the situation of insufficient training data. However, the proposed composite neural network is mainly for the incorporation of solutions of sub-problems, regardless heterogeneous not.

Ensemble learning and transfer learning have their constraints. Ensemble learning ensembles learners that must have the accuracy $> 50\%$ [6]. Transfer learning [5] assumes, for a source-target domain pair, there is an intermediate representation that can be transferred for domain adaptation. Unlike ensemble learning or transfer learning, the proposed composite network framework allows a generic condition with a statistical performance guarantee. In addition, in the literature, some negative effects have been observed, e.g., Zhou et al. [7] pointed out that "many could be better than all" in the typical ensemble settings, and Chen et al. [8] showed the transfer learning has suffered from the "negative transfer" problem. The papers of Džeroski et al. [9] and of Gashler et al. [10] also concluded that an ensemble is not always strictly better than its best component because of the low diversity between members. These facts imply that the claim "the more components, the better performance" may be not always true. The gap of theoretical analysis that supports or opposes the claim motivates this study.

PM2.5 (particulate matter with a diameter less than 2.5 μm) has become a great concern due to its proven threat to human health [11]. PM2.5 is a collection of aerosol material primarily composed of ammonium sulfate, ammonium nitrate, organic carbonaceous mass, elemental carbon, and crustal mineral material emitted from sources such as vehicles, power plants and factories, fossil fuel burning, construction, farming activities, sea salt and dust, and remote transportation [2], [12]–[14]. Both the constituents and sources of PM2.5 vary from one location to the other [12], [13], from one season to the other [2], [14]. For instance, for the seaside rural areas, dust and sea salt are the major causes, while in industrialized countries, fossil fuel burning is the major source. Therefore, PM2.5 prediction must be temporally and spatially dependent. The life cycle and dispersion of PM2.5 depend on factors such as the type of PM2.5, weather conditions, terrain context, and chemical transformations that furthermore complicate the PM2.5 prediction [2]. As a result, predicting the PM2.5 level in the next few hours for a particular area is a great challenge.

In this paper, we answer the challenge of solving complicated applications, and propose a framework and construction algorithms for a composite neural network. Then we use the complicated application – PM2.5 prediction to demonstrate the efficacy of the composite neural network and its applicability to complicated real-world problems. As illustrated in Fig 2, first, a complicated application is decomposed into subtasks, and then some of them are selected as the candidates of pre-trained components. Once the pre-trained components are trained separately or obtained elsewhere, and the topology is decided, then an end-to-end training is performed to construct the final composite neural network.

The contributions in this paper are the following. (1) We propose a framework for the composite neural network, and provide a theoretical analysis of statistical performance guarantee. (2) We provide two heuristic algorithms with alternative composite neural network design principles for performance comparison. (3) We empirically evaluate the performance of composite neural network algorithms and several traditional machine learning methods on PM2.5 prediction data sets; the outcomes support the proposed theory.

The remainder of this paper is organized as follows. We introduce the composite neural network in Section 2, and analyze its performance bounds in Section 3. Section 4 includes several algorithms for composite neural network construction. Section 5 shows intensive evaluations of various composite neural network constructions and traditional machine learning methods, and their comparisons. We discuss related work in Section 6 and the issues discovered during this study in Section 7.

II. CONCEPT OF COMPOSITE NEURAL NETWORK

A typical single-layer neural network can be presented as $f_{\sigma,\mathbf{W}_1}(\mathbf{x}) = w_{1,1}\sigma\left(\sum_{i=1}^{d} w_{0,i}\mathbf{x}_i + w_{0,0}\right) + w_{1,0}$, where $\mathbf{x}$ is the input vector, $\mathbf{W}_1$ is the matrix of weights, and σ is the activation function. In this work we consider differentiable activation functions $\sigma : \mathbb{R} \to \mathbb{R}$ such as the the logistic function $\sigma(z) = 1/(1 + e^{-z})$ and the hyperbolic tangent $\sigma(z) = (e^z - e^{-z})/(e^z + e^{-z})$. If there is no ambiguity on the activation function, the σ function is skipped to simplify notation and the neural network is denoted as $f_{\mathbf{W}}(\mathbf{x})$.

A composite neural network (also termed a composite network) is composed of a set of pre-trained and non-instantiated neural network models that form a directed acyclic graph. For a pre-trained model, its weight matrix $\mathbf{W}_j$ is fixed after its original training process, denoted as f_j to distinguish it from a non-instantiated network. A non-instantiated network is denoted as $f_{\mathbf{W}_j}$; its weights $\mathbf{W}_j$ are not determined until the completion of the training process of the whole composite neural network. Both pre-trained and non-instantiated networks are called components of a composite neural network.

TABLE I
SUMMARY OF NOTATIONS

notation	definition
$\mathbf{W}$	a matrix of weights in a neural network
$\sigma(z)$	activation functions in a neural network
$[N]$	$\{1, ..., N\}$; further, $[K]^+ \triangleq \{0, 1, ..., K\}$
$\{(\mathbf{x}^{(i)}, y^{(i)})\}_{i \in [N]}$	a set of N input-label pairs
$f_{\sigma,\mathbf{W}}(\mathbf{x})$	a neural network defined by σ and $\mathbf{W}$
$\{f_j(\mathbf{x})\}_{j \in [K_1]}$	a set of K_1 pre-trained networks; $K_1 \geq 1$
$\{f_{\mathbf{W}_j}(\mathbf{x})\}_{j \in [K_2]}$	a set of K_2 non-instantiated networks
$\{h_j(\mathbf{x})\}_{j \in [K]}$	$\{f_j(\mathbf{x})\}_{j \in [K_1]} \cup \{f_{\mathbf{W}_j}(\mathbf{x})\}_{j \in [K_2]}$
Θ	a matrix of weights in a composite network
$g_{\Theta}(h_1, ..., h_K)$	an r-layer composite network of h_js by σ, Θ:
	$L_{\Theta_{(r+1)}}\left(\sigma_{(r+1)}\left(\cdots \sigma_{(1)}\left(L_{\Theta_{(0)}}(h_1, ..., h_K)\right)\right)\right)$
$L(\Theta; h_1, ..., h_K)$	$\sum_{j=0}^{K} \theta_j h_j(\mathbf{x})$; $h_0 = 1$, linear combination
$\vec{h}_j$	$(h_j(\mathbf{x}^{(1)}), \cdots, h_j(\mathbf{x}^{(N)}))$; $\vec{h}_0 \triangleq \vec{1}$
$\vec{e}_j$	an unit vector in the standard basis of $\mathbb{R}^{K+1}$
$\mathcal{B}_{K+1}$	$\{\vec{e}_j\}_{j \in [K]^+}$

For a given set of K components $\{h_j(\mathbf{x})\}_{j=1}^{K}$, each component h_j, which can be pre-trained or non-instantiated, has an input vector $\mathbf{x}$ and an output vector $\mathbf{y}_j$. Let h_0 be the constant function 1. Then the linear combination with a bias $\Theta = (\theta_0, \theta_1, \ldots, \theta_K)$ is defined as $L(\Theta; h_1, ..., h_K) = \sum_{j=0}^{K} \theta_j h_j(\mathbf{x})$. When Θ is learned in the training phase, the composite network is denoted as $L_{\Theta}(h_1, ..., h_K)$. To extend the notation further, a neural network with h hidden layers is denoted as $L_{\Theta_{(h+1)}}\left(\sigma_{(h+1)}\left(\cdots \sigma_{(1)}\left(L_{\Theta_{(0)}}(h_1, ..., h_K)\right)\right)\right)$, illustrated as in Fig. 2(c), where the braced number in the subscript indicates the layer number. The components can be in any layer and its output can be fed to any components in the upper layers. Example 1 shows an example composite network.

Example 1. *A composite neural network* $\sigma_{(2)}(\theta_{1,0} + \theta_{1,1}f_4(\mathbf{x}_4) + \theta_{1,2}\sigma_{(1)}(\theta_{0,0} + \theta_{0,1}f_1(\mathbf{x}_1) + \theta_{0,2}f_{\mathbf{W}_2}(\mathbf{x}_2) + \theta_{0,3}f_3(\mathbf{x}_3)))$, *as depicted in Fig. 2(c), can be denoted as* $\sigma_{(2)}\left(L_{(1)}\left(f_4, \sigma_{(1)}\left(L_{(0)}(f_1, f_{\mathbf{W}_2}, f_3)\right)\right)\right)$, *with* Θs *removed for simplicity.*

We assume that the training algorithm of the composite network is the stochastic gradient descent backpropagation algorithm and the loss function is the L_2-norm of the difference vector. The loss function for a trained composite neural network g_{Θ} is defined as

$$\mathcal{E}_{\Theta}(\mathbf{x}; g_{\Theta}) = \frac{\langle g_{\Theta}(\mathbf{x}) - \vec{y}, g_{\Theta}(\mathbf{x}) - \vec{y}\rangle}{N}, \tag{1}$$

where $\langle\cdot, \cdot\rangle$ is the standard inner product and $\vec{y}$ is the ground truth. $\mathcal{E}_{\Theta}(\mathbf{x}; g_{\Theta})$ may be shortened to $\mathcal{E}(g_{\Theta})$. Clearly, the total loss depends on the training data $\mathbf{x}$, the components defined by $\{h_j\}_{j=1}^{K}$, the output activation σ, and the weight vector $\mathbf{W}$. Define $\mathcal{E}(\mathbf{x}; f_j)$ (shortened to $\mathcal{E}(f_j)$, if there is no ambiguity) as the loss function of a single component f_j. It is expected that a good composite network design has low L2 loss, in particular lower than all its pre-trained components. Therefore, the goal is to find a feasible Θ such that it meets the "No-Worse" property, i.e., $\mathcal{E}(g_{\Theta}) < \min_{j \in [K]} \mathcal{E}(f_j)$.

In the following section we will prove that in some reasonable conditions, with high probability, a composite network has strictly lower training L2 loss than all of its pre-trained components. The expectation of L2 loss of a composite network is also with high probability lower than the expectation of the loss of all its pre-trained components. Furthermore, we will show a multi-layer composite network of mixed non-instantiated and pre-trained models that also, with high probability, performs better than any of its pre-trained models.

III. THEORETICAL ANALYSIS

In this section, we first analyze the loss functions of a single-layer composite network, and subsequently extend the analysis to a complicated composite network to explore the characteristics of the composite network. Due to limited space, only ideas

and sketches of proof are presented in this section. For the complete proof, please refer to the appendix in the supplementary material.

A composite network constructed from a given set of pre-trained components $\{f_j\}_{j=1}^K$ forms an acyclic directed graph, which can be represented by postorder tree traversal. Without loss of generality, we assume the dimension of the output vector of all components is 1 in the following proofs. We denote $[K]^+$ the set from 0 to K, $\vec{f}_0 = \vec{1}$, and $\vec{f}_j = (f_j(\mathbf{x}^{(1)}), \cdots, f_j(\mathbf{x}^{(N)}))$ as the sequence of the status of f_j with input data $\mathbf{x}$ during the training phase. Similarly, the representation of the ground truth is $\vec{y} := (y^{(1)}, \cdots, y^{(N)})$. Let $\vec{e}_j$ be an unit vector in the standard basis of $\mathbb{R}^K$ for $j \in [K]$, e.g., $\vec{e}_1 = (1, 0, \cdots, 0)$ and $\mathcal{B}_K := \{\vec{e}_j\}_{j=1}^K$. By C^1-mapping (function) we mean the mapping is differentiable and its derivative is a continuous function.

The following assumptions are the default conditions in the following proofs.

A1. Linearly independent components assumption:
   $\forall i \in [K]^+, \nexists\{\beta_j\} \subset \mathbb{R}$, s.t. $\vec{f}_i = \sum_{j \in [K]\setminus\{i\}} \beta_j \vec{f}_j$.

A2. No perfect component assumption:
   $\min_{j \in [K]} \left\{ \sum_{i \in [N]} |f_j(\mathbf{x}^{(i)}) - \mathbf{y}^{(i)}| \right\} > 0$.

A3. The activation function and its derivative are C^1-mappings (i.e., it is differentiable and its differential is continuous) and the derivative is non-zero at some points in the domain.

A4. The number of components, K, is less than $2\sqrt{N} - 1$, where N is the size of the training data set.

A. Single-Layer Composite Network

The first theorem states that if a single-layer composite network satisfies the above five assumptions, it meets the "No-Worse" property with high probability.

Theorem 1. *Consider a single-layer composite network $g(\mathbf{x}) = L_{(1)}(\sigma(L_{(0)}(f_1, ..., f_K)))(\mathbf{x})$. Then with probability of at least $1 - \frac{K+1}{\sqrt{N}}$ there exists $\mathbf{\Theta} = \{\Theta_1, \Theta_0\}$ s.t. $\mathcal{E}_{\mathbf{\Theta}}(\mathbf{x}; g) < \min_{j \in [K]} \mathcal{E}(f_j(\mathbf{x}))$.*

We discuss two cases of the activation σ.

- Case 1: σ is a linear function.
- Case 2: σ is not a linear function.

(**Case 1**) σ is a linear activation such that a single-layer composite network such as $L_{(1)}(\sigma(L_{(0)}(f_1, ..., f_K)))$ can be rewritten as a linear combination with bias, i.e., $g_\theta(\mathbf{x}) = \sum_{j \in [K]^+} \theta_j f_j(\mathbf{x})$ with a mean squared error of $\mathcal{E}_{\mathbf{\Theta}}(\mathbf{x}; g) = \frac{1}{N} \sum_{i=1}^N (g_{\mathbf{\Theta}}(\mathbf{x}^{(i)}) - \mathbf{y}^{(i)})^2$. Clearly, the composite network g_θ should have a mean squared error equal to or better than any of its components f_j, as g_θ can always act as its best component. To obtain the minimizer $\mathbf{\Theta}^*$ for the error $\mathcal{E}_{\mathbf{\Theta}}(\mathbf{x}; g)$, we must compute the partial differential $\partial \mathcal{E}_{\mathbf{\Theta}} / \partial \theta_j$ for all $j \in [K]^+$. After some calculations [15], we have Eq. (2).

$$\mathbf{\Theta}^* = [\theta_j]_{j \in [K]^+} = \left[\langle \vec{f}_i, \vec{f}_j \rangle \right]_{i,j \in [K]^+}^{-1} \times \left[\langle \vec{f}_i, \vec{y} \rangle \right]_{i \in [K]^+} \tag{2}$$

Since Assumption A1 holds, the inverse matrix $\left[\langle \vec{f}_i, \vec{f}_j \rangle \right]_{i,j \in [K]^+}^{-1}$ exists and can be written down concretely to obtain $\mathbf{\Theta}^*$ as in Eq. (2). Lemma 1 summarizes the above arguments.

Lemma 1. *Set $\mathbf{\Theta}^*$ as in Eq. (2); then*

$$\mathcal{E}(g_{\mathbf{\Theta}^*}) \leq \min_{j \in [K]^+} \{\mathcal{E}(f_j)\}. \tag{3}$$

There is a $\leq$ constraint on the loss function $\mathcal{E}(g_{\mathbf{\Theta}^*})$ in Eq. (3) that is replaced by $<$ and a probability bound. If $\mathbf{\Theta}^*$ is not a unit vector, it is obvious that $\mathcal{E}(g_{\mathbf{\Theta}^*})$ must be less than any $\mathcal{E}(f_j)$. Therefore, we proceed to estimate the probability of $\mathbf{\Theta}^* = \vec{e}_{j^*}$, where $j^* \in [K]^+$.

$$\forall i \in [K]^+, \frac{\partial \mathcal{E}}{\partial \theta_i}\Big|_{\mathbf{\Theta} = \vec{e}_{j^*}} = 2\langle \vec{f}_{j^*} - \vec{y}, \vec{f}_i \rangle \tag{4}$$

Eq. (4) shows the gradient of the error function with respect to θ_i conditioned on $\mathbf{\Theta}^* = \vec{e}_{j^*}$, which is the inner products of the difference between f_{j^*} (the output of $g_{\mathbf{\Theta}^*}$) and the ground truth $\vec{y}$, and the output of each pre-trained component $\vec{f}_i$. When the minimizer $\mathbf{\Theta}^* = \vec{e}_{j^*}$, all the differentials $\frac{\partial \mathcal{E}}{\partial \theta_i}$ must equal zero, i.e., $\langle \vec{f}_{j^*} - \vec{y}, \vec{f}_i \rangle = 0$, or $\vec{f}_{j^*} - \vec{y}$ is perpendicular to $\vec{f}_i$. The following Lemma 2 is an implication from the proof of the Johnson-Lindenstrauss Lemma [16], which states that a randomly sampled unit vector $\vec{v}$ (denoted as $\Pr_{\vec{v} \in \mathbb{R}^N}$) is approximately perpendicular to a given vector $\vec{u}$ with high probability in a high dimensional space..

Lemma 2. *For a large enough N and given $\vec{u} \in \mathbb{R}^N$, there is a constant $c > 0$, s.t. for $\eta = cos^{-1}(1 - c/\sqrt{N})$,*

$$\Pr_{\vec{v} \in \mathbb{R}^N} \left\{ |\angle_{\vec{u}, \vec{v}} - \frac{\pi}{2}| \leq \eta \right\} \geq 1 - \frac{1}{\sqrt{N}} \tag{5}$$

where $\angle_{\vec{u}, \vec{v}}$ is the angle between $\vec{u}$ and $\vec{v}$.

The complement of Eq. (5) is

$$\Pr_{\vec{v} \in \mathbb{R}^N} \left\{ |\angle_{\vec{u},\vec{v}} - \frac{\pi}{2}| > \eta \right\} < \frac{1}{\sqrt{N}} \tag{6}$$

Note that angles $\angle_{\vec{y},\vec{f}}$, $\angle_{\vec{f}-\vec{y},\vec{f}}$, and $\angle_{\vec{f}-\vec{y},-\vec{y}}$ are the three inner angles of the triangle such that $\angle_{\vec{y},\vec{f}} + \angle_{\vec{f}-\vec{y},\vec{f}} + \angle_{\vec{f}-\vec{y},-\vec{y}} = \pi$. From Lemma 2, as $\angle_{\vec{y},\vec{f}}$ is likely a vertical angle (i.e., $\pi/2$), $\angle_{\vec{f}-\vec{y},\vec{f}}$ must be less likely to be a vertical angle, which implies $\Pr\{\langle \vec{f}-\vec{y}, \vec{f} \rangle = 0\} \leq \Pr\{|\angle_{\vec{f}-\vec{y},\vec{f}} - \pi/2| < \eta\}$; thus, $\leq \Pr\{|\angle_{\vec{y},\vec{f}} - \pi/2| > \eta\}$. The following Lemma 3 immediately follows Lemma 2 and Eq. (6).

Lemma 3. *Following Lemma 2, then for given $\vec{y} \in \mathbb{R}^N$,*

$$\Pr_{\vec{f} \in \mathbb{R}^N} \left\{ \langle \vec{f} - \vec{y}, \vec{f} \rangle = 0 \right\} < \frac{1}{\sqrt{N}}.$$

Lemma 3 shows that the probability of the output of one component is perpendicular to the difference between itself and the ground truth. For K components and a bias, Lemma 4 gives a worst bound.

Lemma 4. $\Pr\left\{ \mathcal{E}(g_{\Theta^*}) = \min_{j \in [K]^+}\{\mathcal{E}(f_j)\} \right\} < \frac{K+1}{\sqrt{N}}$, *i.e.,* $\Pr\left\{ \exists \Theta^* : \mathcal{E}(g_{\Theta^*}) < \min_{j \in [K]^+}\{\mathcal{E}(f_j)\} \right\} \geq 1 - \frac{K+1}{\sqrt{N}}$.

(**Case 2**) σ is not a linear function. The idea of the proof is to find an interval in the domain of σ such that the output of $L_{(1)}(\sigma(\cdot))$ approximates a linear function as close as possible. This means there is a setting such that the non-linear activation function performs almost as well as the linear one; since the activation $L_{(1)}(\sigma(\cdot))$ acts like a linear function, the lemmas of Case 1 are applicable. The conclusion of this case is stated as Lemma 7, while we introduce important properties in Lemmas 5 and 6 for key steps in the proof.

Since σ satisfies Assumption A3, the inverse function theorem of Lemma 5 is applicable.

Lemma 5. *(Inverse function theorem [17])*
Suppose μ is a C^1-mapping of an open set $E \subset \mathbb{R}^n$ to $\mathbb{R}^n$, $\mu'(z_0)$ in invertible for some $z_0 \in E$, and $y_0 = \mu(z_0)$. (I.e., μ satisfies Assumption A3.) Then
(1) there exist open sets U and V in $\mathbb{R}^n$ such that $z_0 \in U$, $y_0 \in V$, μ is one-to-one on U, and $\mu(U) = V$;
(2) if ν is the inverse of μ, defined in V by $\nu(\mu(x)) = x$ for $x \in U$, then $\nu \in C^1(V)$.

We also need the following lemma as an important tool.

Lemma 6. *(Taylor's theorem with Lagrange remainder [18])*
If a function $\tau(y)$ has continuous derivatives up to the $(l+1)$-th order on a closed interval containing the two points y_0 and y, then

$$\tau(y) = \tau(y_0) + \tau^{(1)}(y_0)(y - y_0) + \cdots + \frac{\tau^{(l)}(y_0)}{l!}(y - y_0)^l + R_l$$

with the remainder R_l given by the expression for some $c \in [0,1]$:

$$R_l = \frac{\tau^{(l+1)}(c(y - y_0))}{(l+1)!}(y - y_0)^{l+1}.$$

Let $l = 1$, $\tau(y)$ be obtained such that

$$\tau(y) = \tau(y_0) + \tau^{(1)}(y_0)(y - y_0) + \frac{\tau^{(2)}(c(y - y_0))}{2!}(y - y_0)^2. \tag{7}$$

The second-degree term can be used to bound the approximation error.

Now we are ready to give more details to sketch the proof of Case 2. Denote Θ_0^* as the minimizer of Case 1, i.e., the corresponding $g_{\Theta_0^*} = L_{(0)}^*(f_1, ..., f_K)$ satisfies $\mathcal{E}(g_{\Theta_0^*}) < \min_{j \in [K]^+}\{\mathcal{E}(f_j)\} = \mathcal{E}(f_{j^*})$ with high probability, and denote $\Theta_\epsilon = \{\Theta_{1,\epsilon}, \Theta_{0,\epsilon}\}$ corresponding to

$$g_{\Theta_\epsilon} = L_{(1),\epsilon}(\sigma(L_{(0),\epsilon}(f_1, ..., f_K))), \tag{8}$$

called the scaled σ function. Lemma 7 below states a clear condition of a linear approximation of a non-linear activation function.

Lemma 7. *For the given $g_{\Theta_0^*}$, $\{\mathbf{x}^{(i)}\}_{i \in [N]}$, and any $0 < \epsilon \leq 1$, there exists $\Theta_\epsilon = \{\Theta_{1,\epsilon}, \Theta_{0,\epsilon}\}$ such that*

$$\forall i \in [N], |g_{\Theta_\epsilon}(\mathbf{x}^{(i)}) - g_{\Theta_0^*}(\mathbf{x}^{(i)})| < \epsilon. \tag{9}$$

Furthermore, for small enough ϵ,

$$\Pr\left\{ \mathcal{E}(g_{\Theta_\epsilon}) < \min_{j \in [K]^+}\{\mathcal{E}(f_j)\} \right\} \geq 1 - \frac{K+1}{\sqrt{N}}. \tag{10}$$

From the definition of g_{Θ_ϵ}, finding a proper $L_{(0),\epsilon}(\cdot)$ and $L_{(1),\epsilon}(\cdot)$ are the major steps in the proof of Eq. (9). $L_{(0),\epsilon}(\cdot)$ maps the output range of $g_{\Theta_0^*}(\mathbf{x})$ to an interval $(-\gamma + z_0, \gamma + z_0) \subset U_0$ for some $\gamma > 0$ satisfying $\sigma'(z_0) \neq 0$. The scaling factors M_0 and $L_{(0),\epsilon}(\cdot)$ are defined as

$$M_0 = \frac{2}{\gamma} \max_{i \in [N]} \{|g_{\Theta_0^*}(\mathbf{x}^{(i)})|\} \tag{11}$$

$$L_{(0),\epsilon}(\mathbf{x}) = M_0^{-1} g_{\Theta_0^*}(\mathbf{x}) + z_0. \tag{12}$$

It is clear that the range of $L_{(0),\epsilon}(\mathbf{x})$ falls within U_0. $L_{(1),\epsilon}(y)$ intends to map the output range of σ back to $g_{\Theta_0^*}(\cdot)$, and is defined as the expansion of $\tau(\cdot)$ following Eq. (7) without the error term.

$$L_{(1),\epsilon}(y) = M_0 \cdot \tau^{(1)}(y_0) \cdot y + M_0 \cdot \left(z_0 - \tau^{(1)}(y_0) \cdot y_0\right). \tag{13}$$

Reversing the scaling and translating, Eq. (13) can be rewritten as

$$M_0 \left(\tau(y_0) + \tau^{(1)}(y_0)(\sigma\left(M_0^{-1} g_{\Theta_0^*}(\mathbf{x}) + z_0\right) - y_0)\right) - z_0, \tag{14}$$

which equals $g_{\Theta_0^*}(\mathbf{x})$ plus an error bounded by $M_0 M_1 \gamma^2$, where

$$M_1 = 5 \sup_{z \in U_0} \left\{|\tau^{(2)}(\sigma(z) - \sigma(z_0))| \cdot \left(\frac{\sigma(z) - \sigma(z_0)}{z - z_0}\right)^2\right\}. \tag{15}$$

The precise setting of γ can be obtained from $M_0 M_1 \gamma^2 < \epsilon$. Then, with γ and the properties of Lemmas 5 and 6, it can be verified that $g_{\Theta_\epsilon}(\mathbf{x}^{(i)}) = L_{(1),\epsilon}(\sigma(L_{(0),\epsilon}(\mathbf{x}^{(i)})))$ fits Eq. (9).

Eq. (9) implies $(g_{\Theta_\epsilon}(\mathbf{x}^{(i)}) - y^{(i)})^2 < (|g_{\Theta_0^*}(\mathbf{x}^{(i)}) - y^{(i)}| + \epsilon)^2$, which can derive $\mathcal{E}(g_{\Theta_\epsilon}) < \mathcal{E}(g_{\Theta_0^*}) + \Delta(\epsilon)$, where $\Delta(\epsilon)$ is an increasing function of ϵ when the other parameters are fixed. Hence, if ϵ is small enough, we have $\Delta(\epsilon) \leq \frac{\mathcal{E}(f_{j^*}) - \mathcal{E}(g_{\Theta_0^*})}{3}$. By further considering $\mathcal{E}(g_{\Theta_\epsilon}) < \mathcal{E}(g_{\Theta_0^*}) + \Delta(\epsilon)$, it is easy to see that $\mathcal{E}(g_{\Theta_\epsilon}) < \mathcal{E}(f_{j^*})$. The probability of $\mathcal{E}(g_{\Theta_0^*}) < \mathcal{E}(f_{j^*})$ of Eq. (10) can be inferred from Lemma 4 of Case 1. Example 2 below shows how to construct a scaled activation function that satisfies Eq. (9).

Example 2. *Here we take a logistic function $\sigma(z) = \frac{1}{1+e^{-z}}$ in the context of PM2.5 prediction to construct a scaled logistic function. Let notations $g_{\Theta_0^*}(\cdot)$, z_0, U_0, V_0, and $\tau(\cdot)$ be as previously defined. The assumption that the highest PM2.5 measurement is less than 1000 (i.e., $\max_{i \in [N]}\{|g_{\Theta_0^*}(\mathbf{x}^{(i)})|\} < 1000$) fits the reality for most countries. Observe that $\sigma^{(1)}(0) = \frac{1}{4}$, $\sigma(0) = \frac{1}{2}$, and hence it is valid to set $z_0 = 0$. Consider $(-\gamma, \gamma) \subset [-1, 1]$ and hence , $y_0 = \sigma(0)$ and $y = \sigma(z) \in (0.25, 0.75)$.*

The inverse function of $\sigma(z)$ is $\tau(y) = \ln \frac{y}{1-y}$ for $y \in (0, 1)$, which also can be represented as $\tau(y) = 4y - 2 + \frac{\tau^{(2)}(c(y-y_0))}{2}(y - y_0)^2$ for some $c \in (0, 1)$ by Lemma 6. From Eq. (11), the scaling factors $M_0 = 2\gamma^{-1}\max_{i \in [N]}\{|g_{\Theta_0^*}(\mathbf{x}^{(i)})|\} < 2 \cdot 10^3 \gamma^{-1}$, and from Eq. (15), $M_1 = 5\sup_{z \in U_0}\left\{\tau^{(2)}(\sigma(z) - \sigma(z_0))\left[(\sigma(z) - \sigma(z_0))/(z - z_0)\right]^2\right\}$, which is less than 50 for $z \in (-\gamma, \gamma)$. From Eq. (14), the scaled logistic function as $g_{\Theta_\epsilon}(\mathbf{x}) = M_0 \cdot \left(4\sigma\left(M_0^{-1} g_{\Theta_0^*}(\mathbf{x})\right) - 2\right)$.

Now we claim that for any given $\epsilon \in (0, 1]$, $g_{\Theta_0^*}(\cdot)$ and $\{\mathbf{x}^{(i)}\}_{i \in [N]}$, we have $|g_{\Theta_\epsilon}(\mathbf{x}^{(i)}) - g_{\Theta_0^*}(\mathbf{x}^{(i)})| < \epsilon$. Here is a short verification. Observe $\forall i \in [N], M_0^{-1} g_{\Theta_0^*}(\mathbf{x}^{(i)}) \in (-\gamma, \gamma)$. Also, if $z \in (-\gamma, \gamma)$, then $|\frac{\tau^{(2)}(c(y-y_0))}{2}|(y - y_0)^2 < M_1\gamma^2$. Recall that $\tau \circ \sigma(\cdot)$ is an identity function, $y = \sigma(M^{-1} g_{\Theta_0^*}(\mathbf{x}))$, and $|\tau(y) - (4y - 2)| < M_1\gamma^2$. That is, $|M_0^{-1} g_{\Theta_0^*}(\mathbf{x}) - \left[4\sigma(M_0^{-1} g_{\Theta_0^*}(\mathbf{x})) - 2\right]| < M_1\gamma^2$. Multiply by M_0 on both sides and replace the bracket term with $g_{\Theta_\epsilon}(\mathbf{x})$; we have $|g_{\Theta_0^*}(\mathbf{x}) - g_{\Theta_\epsilon}(\mathbf{x})| < M_0 M_1\gamma^2 < 10^5\gamma$. Hence, setting $\gamma = 10^{-5}\epsilon$ verifies this claim.

From Lemma 7, we can conclude that there exists Θ_ϵ such that a non-linear single-layer composite network performs at least as well as the linear case with arbitrary small error. Thus, the proof of Case 2 is concluded. The proofs of Cases 1 and 2 above complete the proof of Theorem 1.

B. Complicated Composite Network

In the previous section, we investigated the performance of a single-layer composite network comprising several pre-trained components connected by an activation function. Now we consider expanding the composite network in terms of width and depth. Formally, for a given pre-trained component f_K and a trained composite network g_{K-1} of $K - 1$ components $(f_1, ..., f_{K-1})$, we study the following two questions in this section.

Q1: (Adding width) By adding a new pre-trained component f_K, we define $g_K = L_{(1)}(\sigma(L_{(0)}(f_1, ..., f_{K-1}, f_K))$. Is there Θ such that $\mathcal{E}(g_{K-1}) > \mathcal{E}_\Theta(g_K)$?

Q2: (Adding depth) By adding a new pre-trained component f_K, let $g_K = L_{(K)}(\sigma(L_{(K-1)}(g_{K-1}, f_K))$. Is there Θ such that $\mathcal{E}(g_{K-1}) > \mathcal{E}_\Theta(g_K)$?

Lemma 8 answers Q1, and we require Proposition 1 as the base of induction to prove it.

Lemma 8. *Set $g_K = L_{(1)}(\sigma(L_{(0)}((f_1, ..., f_{K-1}, f_K))))$. With probability of at least $1 - \frac{K+1}{\sqrt{N}}$, there is Θ s.t. $\mathcal{E}(g_{K-1}) > \mathcal{E}_\Theta(g_K)$.*

Proposition 1. *Consider the case of only two pre-trained models f_0 and f_1. There exists $(\alpha_0, \alpha_1) \in \mathbb{R}^2$ s.t.*

$$\sum_{i \in [N]} (f_1(\mathbf{x}^{(i)}) - y^{(i)})^2 > \sum_{i \in [N]} \left(\alpha_0 f_0(\mathbf{x}^{(i)}) + \alpha_1 f_1(\mathbf{x}^{(i)}) - y^{(i)} \right)^2$$

with a probability of at least $1 - \frac{2}{\sqrt{N}}$.

Proposition 1 can be proved by solving the inequality directly for the case of $K = 2$, and then generalizing the result to larger K by induction with the help of Lemma 3 to prove Lemma 8. Adding a new component f_K to a composite network g_{K-1} as in Q2, the depth of resulting g_K increments by 1. If $\vec{g}_{K-1}$ and $\vec{f}_K$ satisfy A1 and A2, consider $\{g_{K-1}, f_K\}$ as a new set of $\{f_1, f_2\}$ in the same layer. Consequently, we can apply the arguments in Case 2 of Theorem 1 to show Lemma 9 in the following, which answers Q2 and says the resulting g_K has a minimizer Θ^* such that with high probability the loss decreases.

Lemma 9. *Set $g_K = L_{(1)}(\sigma(L_{(0)}((g_{K-1}, f_K)))$. If $\vec{g}_{K-1}$ and $\vec{f}_K$ satisfy A1 and A2, then with a probability of at least $1 - \frac{2}{\sqrt{N}}$, there is Θ s.t. $\mathcal{E}(g_{K-1}) > \mathcal{E}_\Theta(g_K)$.*

The proof of Lemma 9 is similar to the proof of Case 2 in the previous sub-section. Lemmas 8 and 9 imply a greedy strategy to build a complicated composite network. Recursively applying both lemmas, we can build a complicated composite network as desired. Theorem 2 gives a formal statement of the constructed complicated composite network with a probability bound. The proof of Theorem 2 is based on mathematical induction on layers and the worst case probability is over-estimated by assuming each layer could have up to K components.

Theorem 2. *For an H-hidden layer composite network with K pre-trained components, there exists Θ^* s.t.*

$$\mathcal{E}_{\Theta^*}(g) < \min_{j \in [K]^+} \{\mathcal{E}(f_j)\}$$

with a probability of at least $\left(1 - \frac{K+1}{\sqrt{N}}\right)^H$.

IV. COMPOSITE NETWORK CONSTRUCTION

The theoretical analysis in the previous section suggests that with high probability, a trained composite network performs better than any of its pre-trained components. It also encourages users to apply their domain expertise to design and train critical pre-trained components and incorporate them in their composite network. In this section, we propose heuristic algorithms for composite network construction. Ensemble learning is a simple case of the composite network that will be evaluated and compared with the proposed algorithm.

For a given set of components, we define the component whose output gives an answer to the main problem as a base component. If the outputs of a component do not directly answer the main problem, we call this an auxiliary component. For example, in the problem of PM2.5 value prediction, the base components output their PM2.5 predictions, whereas a component predicting weather conditions such as wind speed and precipitation is categorized as an auxiliary component.

The Deep Binary Composite Network (DBCN) Algorithm depicted in Algorithm 1 is a greedy method, the main idea of which is to construct a composite network by inserting one component at a time in some particular order. After each insertion, the depth of the network is increased by 1, as described in Lemma 9. We consider the base components first in the insertion order since a base component answers the main problem and it makes sense to use auxiliary components to enhance the performance of the base components later. The pre-trained components are considered before the non-instantiated ones, as pre-trained components are commonly well-crafted and performance-proven. Thus, we insert the components such that pre-trained components are ahead of non-instantiated components, and for each pre-trained and non-instantiated set, the base components are ahead of auxiliary components; finally, the components with lower L2 errors are before those with higher L2 errors.

Algorithm 1 takes pre-trained components $\{f_j\}_1^{K_1}$ and non-instantiated components $\{f_{W_j}\}_{K_1+1}^K$, sorted according to the criteria in the previous paragraph, as inputs, and outputs a deep binary composite network. Line 1 initializes the variables used in this algorithm. The first-level **for** block (from Lines 2 to 12) computes the composite network g_j of depth j, iteratively. The second-level **for** block from Lines 3 to 9 generates possible composite networks with both linear and modified logistic activation functions $\sigma(\cdot)$. In Line 10, we use traditional stochastic gradient descent backpropagation to train every composite network in $\mathcal{T}_j$. Line 12 finds the composite network with the lowest L2 error. Lines 13 to 20 prune the obtained $\{g_j\}$ to avoid over-fitting. Once the L2 loss gain is larger than a specified pruning threshold Δ, the pruning process stops and the algorithm outputs the current g_j; otherwise, g_{j-1} is examined as a consequence.

The second algorithm, Balanced Base Composite Network (BBCN), is presented in Algorithm 2. The first-level **for** block (from Lines 4 to 16) generates a flat composite network from the base components, in which each iteration constructs a level of the composite network. The **for** block (Lines 5 to 15) combines a pair of two base components or two subtrees. Line 18

Algorithm 1: Deep Binary Composite Network

Input: $\mathcal{F} = \{f_j\}_0^{K_1} \cup \{f_{W_j}\}_{K_1+1}^{K}$, a set of activation functions $\mathcal{A}$, pruning threshold Δ
Output: g_K

1   $g_1 \leftarrow f_1; \forall j \leq K, \mathcal{T}_j \leftarrow \emptyset$
2   **for** $j = 2$ *to* K **do**
3   **for** $\sigma(\cdot) \in \mathcal{A}$ **do**
4   **if** $j \leq K_1$ **then**
5              $\mathcal{T}_j \leftarrow \mathcal{T}_j \cup \{\sigma(g_{j-1}, f_j)\}$
6   **else**
7              $\mathcal{T}_j \leftarrow \mathcal{T}_j \cup \{\sigma(g_{j-1}, f_{W_j})\}$
8   **end**
9   **end**
10   Train all $h \in \mathcal{T}_j$
11     $g_j \leftarrow \mathsf{argmin}_{h \in \mathcal{T}_j}\{\mathcal{E}(h)\}$
12   **end**
13   **for** $j = K$ *to* 2 **do**
14   **if** $\mathcal{E}(g_j) - \mathcal{E}(g_{j-1}) \leq \Delta$ **then**
15        $g_j \leftarrow g_{j-1}$
16   **else**
17   output g_j
18   break
19   **end**
20   **end**

Algorithm 2: Balanced Base Composite Network

Input: $\mathcal{F} = \{f_j\}_0^{K_1} \cup \{f_{W_j}\}_{K_1+1}^{K}$, a set of activation functions $\mathcal{A}$, the number of base components K_0, pruning threshold Δ
Output: g_K

1   $\forall j \in [K_0], h_{0,j} \leftarrow f_j$
2   $\forall s \leq \lceil \log_2(K_0) \rceil, t \leq \lceil K_0/2^s \rceil, \mathcal{T}_{s,t} \leftarrow \emptyset$
3   $\forall j \leq K, \mathcal{T}_j \leftarrow \emptyset$
4   **for** $s = 1$ *to* $\lceil \log_2(K_0) \rceil$ **do**
5   **for** $t = 1$ *to* $\lceil K_0/2^s \rceil$ **do**
6   **if** $\lceil K_0/2^{s-1} \rceil$ *is an odd number* & $t = \lceil K_0/2^s \rceil$ **then**
7           $h_{s,t} \leftarrow h_{s-1,2t-1}$;
8   **else**
9   **for** $\sigma(\cdot) \in \mathcal{A}$ **do**
10             $\mathcal{T}_{s,t} \leftarrow \mathcal{T}_{s,t} \cup \{\sigma(h_{s-1,2t-1}, h_{s-1,2t})\}$;
11   **end**
12   Train all $h \in \mathcal{T}_{s,t}$;
13           $h_{s,t} \leftarrow \mathsf{argmin}_{h \in \mathcal{T}_{s,t}}\{\mathcal{E}(h)\}$;
14   **end**
15   **end**
16   **end**
17   $g_{K_0} \leftarrow h_{\lceil \log_2(K_0) \rceil, 1}$
18   Run Algorithm 1 on $(\{g_{K_0}\} \cup \mathcal{F} \setminus \{f_j\}_{j \in [K_0]})$

---

calls Algorithm 1 to complete the execution. In general, Algorithm 1 generates a deep binary composite network, whereas Algorithm 2 constructs a more balanced composite network, as shown in Fig. 4.

V. PM2.5 PREDICTIONS

In this section, we design five pre-trained components and a non-instantiated component and apply composite network construction methods including exhaustive search, ensemble learning [6], and Algorithm 1 (DBCN) and Algorithm 2 (BBCN) for PM2.5 prediction. Real-world open data was used to numerically compare the performance of different construction methods and to examine the correctness and efficacy of the proposed theory. In addition, we also compared the methods with traditional machine learning methods, namely, SVM [19] and random forests [20]. For the hardware and software environment, each of the three servers used in this evaluation was equipped with two Intel Xeon CPUs, 128GB memory, four NVIDIA 1080 GPUs, the Linux operating system, and Keras and Tensorflow as deep learning platforms.

A. Datasets

The open data were from two sources: the Environmental Protection Administration (EPA) for air quality data [21], and the Central Weather Bureau (CWB) for weather data [22]. There are 21 features in the EPA dataset including values such as PM2.5, PM10, SO_2, CO, NO, and NO_x. The EPA air quality data were collected from eighteen monitoring stations recorded hourly. The second dataset, the CWB open data, has one record per six hours, collected from 31 monitoring stations with 26

features, including temperature, dew point, precipitation, and wind speed and direction. In this study, for all evaluations, the data of years 2014 and 2015 were used as training data and those of 2016 as testing data.

We created a grid of $30 \times 38 = 1140$ km^2 covering the Taipei area, each block of which was 1×1 km^2. The EPA and CWB data were loaded into the corresponding blocks so that both datasets were temporally aligned at the hour scale (i.e., one record per hour). Interpolation was applied to the CWB data to downscale from 6 hours to 1 hour. Note that there were 1140 blocks in the grid, whereas there were only 18 EPA stations and 31 CWB stations; thus more than 1000 blocks were empty, i.e. without EPA or CWB data. We adopted the KNN method ($K = 4$, i.e., averaging the values of the four nearest neighbors) to initialize the values of the empty blocks, as discussed in [23].

B. Pre-trained Component Design

Here we introduce the design rationales of the five pre-trained components in this evaluation. As PM2.5 dispersion is highly spatially and temporally dependent, we designed four pre-trained components as base components to model this dependency. Among these, two were convolutional LSTM neural networks (ConvLSTMs [24]) with the EPA data (denoted as f_1) and CWB data (denoted as f_2) as input; the other two were fully connected neural networks (FNNs) with the EPA data (denoted as f_3) and CWB data (denoted as f_4) as input. To model the temporal relationship conveniently using the neural network, the data was fed to the pre-trained components one sequence at a time. We used two pairs of components—f_1 and f_2, and f_3 and f_4—for the same functions to determine whether component redundancy improves performance. The fifth pre-trained component (denoted as f_5) was to model the association between time and the PM2.5 value.

TABLE II
LSTM (LsM) v.s. ConvLSTM (CvL)

Hour	Dataset	EPA		CWB	
	Models	Training	Testing	Training	Testing
+24h	LsM	8.4158	10.9586	8.2741	11.3947
	CvL	7.5873	**10.5789**	8.5529	**11.2074**
+48h	LsM	8.7185	11.5229	8.5232	11.8144
	CvL	8.6541	**11.3904**	8.2890	**11.7081**
+72h	LsM	8.7530	11.7329	8.8905	11.8672
	CvL	8.8170	**11.5279**	9.2177	**11.7756**

The first experiment was designed to examine the effect of the grid structure in capturing the spatial relationship by comparing the outcomes of LSTM and ConvLSTM. The LSTM model only used the EPA and CWB data without spatial information about the monitoring stations, whereas the ConvLSTM model used the grid data (i.e., considering the whole 1140 blocks with KNN ($K = 4$) initialization). The accuracy of both models measured in RMSE is presented in Table II, which shows the ConvLSTM performs consistently better for the +24h (next 24 hours), +48h (next 48 hours), and +72h (next 72 hours) predictions. Hence, we selected ConvLSTM as the model for f_1 and f_2.

TABLE III
Various configurations of pre-trained components

Model	Forecast Train.Params	+24h		+48h		+72h	
		Training	Testing	Training	Testing	Training	Testing
f_1	917492	7.5873	**10.5789**	8.6541	**11.3904**	8.8170	**11.5279**
$f_{1,Wr}$	3632482	9.3054	11.9440	9.1503	11.6550	8.1616	11.7556
$f_{1,Dr}$	1278692	7.6342	10.9471	8.6297	11.4844	9.0803	11.5993
f_2	916908	8.5529	**11.2074**	8.2890	**11.7081**	9.2177	**11.7756**
$f_{2,Wr}$	3631322	7.0685	11.4974	9.2233	12.0710	9.1766	11.9827
$f_{2,Dr}$	790828	6.5404	11.7970	8.4491	8.4491	9.1500	11.9162
$f_{3,(2)}$	1038054	11.6064	**10.8907**	11.9008	**11.6977**	12.1729	11.9999
$f_{3,(3)}$	1068538	11.5648	10.9179	11.9726	11.7017	12.0585	**11.9414**
$f_{4,(2)}$	582038	11.8238	11.3400	11.6948	**11.6147**	11.9484	11.8687
$f_{4,(3)}$	603318	11.8253	**11.2748**	11.7112	11.6176	12.0199	**11.7512**

In the second experiment, we trained the four pre-trained components (f_1, f_2, f_3, f_4) individually with different configurations. For instance, we trained the ConvLSTM models (f_1 and f_2) with a normal configuration, a deeper one (denoted as Dr) with stack of two LSTMs, and a wider one (denoted as Wr) with a double-width ConvLSTM. Similarly, FNN models f_3 and f_4 were trained with two or three hidden layers, (denoted as $f_{i,(2or3)}$). Their performance was measured by RMSE as shown in Table III. The best performing configurations were selected for the pre-trained components in the following experiments.

Note that instead of using execution time as a measurement of time complexity, we indicated the complexity using the number of trainable (tunable) parameters in our study, as shown in the second column of Table III, as the execution times varied widely even for the same training configuration due to diverse server execution contexts, randomness incurred from training commands, and hyperparameter tuning setups.
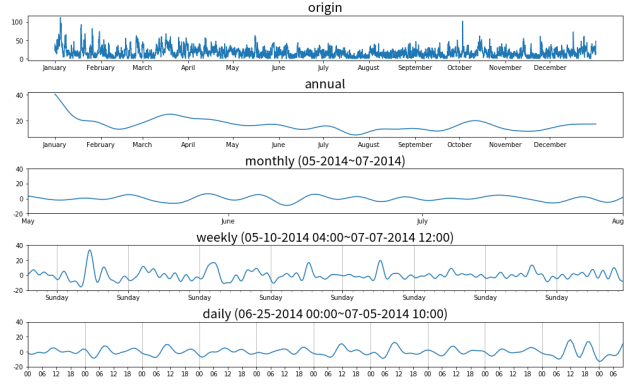
Fig. 3. Annual PM2.5 values at different frequencies

The fifth pre-trained component (f_5) is the association between time and PM2.5 value, which is highly temporally dependent.

Fig. 3 shows the PM2.5 values resulting from the different frequency filters [25]. The top figure shows the original PM2.5 values of the Taitung EPA station in 2014 and the second figure shows the annual trend, which clearly shows that cold months are prone to high PM2.5 pollution. The third graph shows the PM2.5 trend from May to July, which does not reveal a consistent pattern. The fourth figure shows the trends within a week: we observe lower PM2.5 values during the weekend. The fifth figure is the daily trend: PM2.5 values are lower after midnight. Based on these observations, we generated an embedding [26] of features including the month, day of the week, and the hour of the day, and trained a LSTM model labeled with PM2.5 values as the pre-trained component f_5.

C. Composite Network

TABLE IV
PRE-TRAINED COMPONENTS AND TESTING RMSE

Component	Data	+24h	+48h	+72h
f_1: ConvLSTM (2 CNN layers, 1 LSTM)	EPA	10.5789	11.3904	11.5279
f_2: ConvLSTM (2 CNN layers, 1 LSTM)	CWB	11.2074	11.7081	11.7756
f_3: FNN (2 hidden layers)	EPA	10.6459	11.3291	11.6169
f_4: FNN (2 hidden layers)	CWB	11.5112	11.6915	11.8017
f_5: LSTM	hr-week-month	11.4738	11.5359	11.4540

EPA 9 features: CO, NO, NO2, NOx, O3, PM10, PM2.5, SO2, THC
CWB 5 features: AMB-TEMP, RH, rainfall, wind direction-speed (represented as a vector)

There are five pre-trained components from f_1 to f_5 and one non-instantiated auxiliary component, denoted as f_{W_6}, for the composite network construction. The model of f_{W_6} is a convolutional neural network (CNN) with CWB weather data and forecasts as input to predict upcoming precipitation. The six components are connected by activation functions, either a linear function or a scaled logistic function ($S(z) = 2000/(1 + e^{-z/500}) - 1000$). Note that any activation function that meets all six assumptions in Sec. 3 could be used; for simplicity, we used only the scaled logistic function. The prediction accuracy in RMSE of all five pre-trained components is listed in Table IV. Note that in this study we did not set out to design an optimized composite network for the best PM2.5 prediction. Rather, our main purpose was to implement and evaluate the proposed composite network theory. Nevertheless, the design of components and composite network follows the advice of domain experts and exhibits reasonably good performance in PM2.5 prediction.

1) DBCN and BBCN: The step-by-step running of Algorithm 1 (DBCN) and the results are shown in Table V for the +24h predictions. First, f_1 is automatically selected as g_1, after which f_3 is included, as it has the lowest RMSE among the remaining components. In the first column of the table, $L(g_1, f_3)$ has a lower RMSE than $SL(g_1, f_3)$ and is selected as g_2, as marked in the last column ("Front-runner"). (Note that SL is an abbreviation of the scaled logistic function cascading a linear function.) Next, Algorithm 1 generates the composite network $L(g_5, f_{W_6})$ with a testing RMSE of 10.9531 for the +24h prediction. Table VI shows the +48h and +72h prediction results: the generated models are different from each other and the model for +24h.

The "Trainable/total" column indicates the number of trainable parameters and total parameters during the training phase. The trainable parameters are updated during each backpropagation stochastic gradient descent optimization, and the total parameters are the number of trainable parameters plus the fixed parameters in the pre-trained components. As only the trainable parameters are updated during training, the composite network framework may greatly alleviate many burdens in training a complicated composite network.

The processes and results of Algorithm 2 (BBCN) are shown in Table VII for the +24h PM2.5 predictions and in Table VIII +48h and +72h. Note that Algorithm 2 constructs a composite network by merging the base components in the beginning:

TABLE V
COMPOSITE NETWORKS USING ALGO 1: DBCN, +24H

| Model | RMSE | | Parameters | |
	Training	**Testing**	Trainable/total	Front-runner
$g_1 \leftarrow f_1$			0/	
$L(g_1, f_3)$	7.2128	**10.2277**	666/1956230	g_2
$SL(g_1, f_3)$	7.3311	10.3454	666/1956230	
$L(g_2, f_2)$	7.1364	10.2410	666/2873814	
$SL(g_2, f_2)$	7.3208	**10.2409**	666/2873814	g_3
$L(g_3, f_4)$	7.0787	**10.3039**	666/3456518	g_4
$SL(g_3, f_4)$	7.1931	10.3501	666/3456518	
$L(g_4, f_5)$	7.0911	10.3275	666/4411100	
$SL(g_4, f_5)$	7.0560	**10.2119**	666/4411100	g_5
$L(g_5, f_{W_6})$	6.9608	10.1131	42046/4453146	
$SL(g_5, f_{W_6})$	6.9705	**10.1053**	42046/4453146	g_6

TABLE VI
COMPOSITE NETWORKS USING ALGO 1: DBCN, +48H, +72H

| Prediction | Model | RMSE | | Parameters | |
		Training	**Testing**	Trainable/total	Front-runner
+48h	$SL(g_4, f_2)$	8.0678	**11.0469**	666/4411100	g_5
	$SL(g_5, f_{W_6})$	7.8941	**10.9531**	42046/4453146	g_6
+72h	$L(g_4, f_3)$	8.2305	**11.4274**	666/4411100	g_5
	$L(g_5, f_{W_6})$	8.2448	**11.2541**	42046/4453146	g_6

the first row of Table VII combines f_1 and f_2, and the second row combines f_3 and f_4. Generally, both DBCN and BBCN methods meet the claim of the proposed composite network theory: combining more pre-trained components yields improved RMSE results. The composite networks constructed using Algorithms 1 and 2 for +24h prediction are contrasted in Fig. 4.

2) Exhaustive Search Construction: In this subsection, an exhaustive search method based on Algorithm 2 is introduced to construct a high-accuracy PM2.5 prediction composite network for use as a high-mark benchmark for comparison. In contrast to the previous approaches, in the exhaustive search approach the parameters inside a pre-trained component can be either fixed or open in order to guarantee the best construction. Hence, instead of the 5 pre-trained and 1 non-instantiated components used by the previous algorithms, we now have five additional pre-trained components with open (tunable) parameters (i.e., non-instantiated components). The new notation $\times$ denotes pre-trained components and $\circ$ denotes non-instantiated components. For instance, f_1° is component 1 but non-instantiated. Inherently, with exhaustive search the construction takes a substantially longer time to complete (i.e., with time complexity of $O(2^K)$), but has the potential for better performance. A complete exhaustive search example for PM2.5 prediction is conducted to evaluate the performance improvement.

For +24h prediction, the the exhaustive search algorithm employs the same composite network layout as Algorithm 2. The best composition combining f_1 and f_2 is $g_1 = SL(f_1^\circ, f_2^\circ)$, as shown in Table IX, which corresponds to combining non-instantiated f_1 and f_2 and applying the scaled logistic activation function results in the lowest RMSE. In the next step, f_3 and f_4 are combined with the front-runner as $g_2 = SL(f_3^\circ, f_4^\circ)$ as shown in Table X. Step 3 considers all possible combinations of g_1 and g_2 to find the best g_3, as shown in Table XI. Note that we treat g_i° as having all non-instantiated components; for $g_i^\times$, all components are pre-trained.

Now only f_5 and f_{W_6} are not combined. Here we examine different sequences of f_5 and f_{W_6}. In Steps 4a and 5a, f_5 is considered first and then f_{W_6}. The results are shown in Table XII. Steps 4b and 5b consider the opposite sequence from the results listed in Table XIII. The best models of g_{5a} and g_{5b} are illustrated in Fig. 5. The composite networks for +48h and
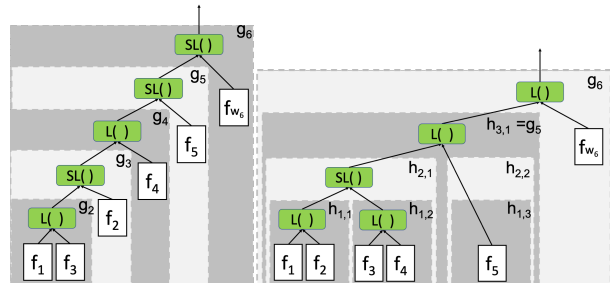

Fig. 4. Composite networks using Algorithms 1 (left) and 2 (right) for +24h prediction

TABLE VII
COMPOSITE NETWORKS USING ALGO 2: BBCN, +24H

Model	RMSE		Parameters	
	Training	**Testing**	Trainable/total	Front-runner
$h_{0,1} \leftarrow f_1, h_{0,2} \leftarrow f_2$				
$L(h_{0,1}, h_{0,2})$	7.1016	**10.4075**	666/1835094	$h_{1,1}$
$SL(h_{0,1}, h_{0,2})$	6.5801	10.4581	666/1835094	
$h_{0,3} \leftarrow f_3, h_{0,4} \leftarrow f_4$				
$L(h_{0,3}, h_{0,4})$	11.4359	**10.7670**	666/1620758	$h_{1,2}$
$SL(h_{0,3}, h_{0,4})$	11.5389	10.8508	666/1620758	
$L(h_{1,1}, h_{1,2})$	7.2375	10.4536	666/3456518	$h_{2,1}$
$SL(h_{1,1}, h_{1,2})$	7.2523	**10.3226**	666/3456518	
$h_{2,2} \leftarrow h_{1,3} \leftarrow h_{0,5} \leftarrow f_5$				
$L(h_{2,1}, h_{2,2})$	7.1069	**10.4712**	666/4411100	$h_{3,1}$
$SL(h_{2,1}, h_{2,2})$	7.1202	10.5064	666/4411100	
$g_5 \leftarrow h_{3,1}$				
$L(g_5, f_{W_6})$	6.9828	**10.1938**	42046/4453146	g_6
$SL(g_5, f_{W_6})$	6.9964	10.2257	42046/4453146	

TABLE VIII
COMPOSITE NETWORKS USING ALGO 2: BBCN, +48H, +72H

Prediction	Model	RMSE		Parameters	
		Training	**Testing**	Trainable/total	Front-runner
+48h	$SL(h_{2,1}, h_{2,2})$	7.9949	**11.0516**	666/4411100	$h_{3,1}$
	$L(g_5, f_{W_6})$	8.5736	**11.0182**	42046/4453146	g_6
+72h	$L(h_{2,1}, h_{2,2})$	8.4460	**11.5100**	666/4411100	$h_{3,1}$
	$L(g_5, f_{W_6})$	9.1848	**11.4153**	42046/4453146	g_6

+72h predictions using exhaustive search were conducted accordingly and their results are used for performance comparisons in the next subsection.

3) Comparisons of All Methods: In this section, we compare the performance of different composite network algorithms, including DBCN, BBCN, exhaustive search, and ensemble methods, as well as machine learning methods, SVM and random forest. In addition, we use Relu and logistic activation functions to replace the scaled logistic function in DBCN and BBCN to show the performance differences. As claimed, the composite network theory guarantees, with high probability, that the composite network has lower RMSE than any of its components, which is supported by all DBCN, BBCN, exhaustive search, and ensemble methods.

We summarize the results of all methods in Table XIV for RMSE, and in Table XX for MAE (mean absolute error) and SMAPE (symmetric mean absolute percentage error). For the SVM and random forest experiments, we used the tools from scikit-learn [27] with pre-trained components only (i.e., f_1 to f_5, with α the total parameters inside these five components.) Likewise with ensemble learning and with ensemble learning with the scaled logistic function as the activation function (denoted as *SL*(ensemble)). The four evaluations yielded close testing RMSE values for all predictions, but the ensemble learning method performed slightly better, while the random forest method seems overfitted, as the training RMSE is low. DBCN performs slightly better than DBNN, and the exhaustive search has the best outcome. For the activation functions, it is interesting to discover that the scaled logistic function performs almost better than the regular logistic and Relu functions.

Now that f_{W_6} is included in composite network construction, it can be seen that DBCN, DBNN, Exhaustive Search (a), and Exhaustive Search (b), as depicted in Fig. 5, show improvements over the composite networks without f_{W_6}. The sum of parameters inside these six components is denoted as β in Table XIV. The second column of the table gives the number of trainable parameters for each evaluation; this shows that for the composite network the training parameters are moderate. Table XX shows the MAE measurements of the evaluations in Table XIV. The ordering of the testing MAE results are very
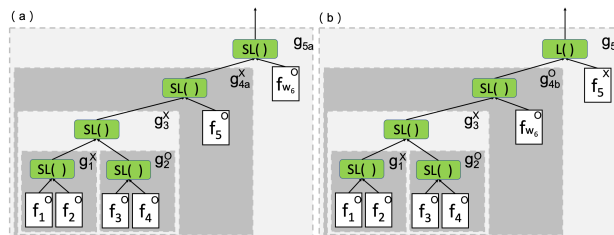

Fig. 5. Composite networks of (a) Tables XII and (b) XIII for +24h prediction

TABLE IX
COMPOSITE NETWORKS OF f_1, f_2 USING EXHAUSTIVE SEARCH, +24H

Step	Model	Training	Testing	Front-runner
1	$L(f_1^\times, f_2^\times)$	7.1016	10.4075	
	$L(f_1^\times, f_2^\circ)$	6.5417	10.2097	
	$L(f_1^\circ, f_2^\times)$	6.6574	10.3484	
	$L(f_1^\circ, f_2^\circ)$	6.3394	10.0423	
	$SL(f_1^\times, f_2^\times)$	6.5801	10.4581	
	$SL(f_1^\times, f_2^\circ)$	6.6648	9.9048	
	$SL(f_1^\circ, f_2^\times)$	6.5052	10.1654	
	$SL(f_1^\circ, f_2^\circ)$	6.5109	**9.7275**	g_1

TABLE X
COMPOSITE NETWORKS OF f_3, f_4 USING EXHAUSTIVE SEARCH, +24H

Step	Model	Training	Testing	Front-runner
2	$L(f_3^\times, f_4^\times)$	11.4359	10.7670	
	$L(f_3^\times, f_4^\circ)$	10.9690	10.8618	
	$L(f_3^\circ, f_4^\times)$	11.0442	10.8285	
	$L(f_3^\circ, f_4^\circ)$	11.2553	10.7017	
	$SL(f_3^\times, f_4^\times)$	11.5389	10.8508	
	$SL(f_3^\times, f_4^\circ)$	11.2916	10.9000	
	$SL(f_3^\circ, f_4^\times)$	11.1600	10.8505	
	$SL(f_3^\circ, f_4^\circ)$	11.1543	**10.6877**	g_2

similar to that of the RMSE results.

VI. RELATED WORK

In this section, we discuss related work in the literature from the perspective of the composite network framework and PM2.5 prediction. For the framework, the composite network is related to the methods such as ensemble learning [6], transfer learning [28] and model reuse [29], [30]. We will also discuss some representative work on air quality prediction.

Ensemble Learning. Typical ensemble learning methods include bagging, boosting, stacking, and linear combination/regression. Since the bagging groups data by sampling and boosting tunes the probability of data [7], these frameworks are not similar to composite neural networks. However, there are fine research results that are instructive for accuracy improvement [7], [9], [10]. In this work, we consider the neural network composition, but not data enrichment.

Among the ensemble methods, stacking is closely related to our framework. The idea of stacked generalization [31], in Wolpert's terminology, is to combine two levels of generalizers. The original data are taken by several level-0 generalizers, after which their outputs are concatenated as an input vector to the level-1 generalizer. According to the empirical study of Ting and Witten [32], the probability distribution of the outputs from level 0, instead of their values, is critical to accuracy. Their experimental results also imply that multi-linear regression is the best level-1 generalizer, and a non-negative weight restriction is necessary for regression but not for classification. However, our analysis shows that the activation functions that satisfy Assumption A3 have a high probability guarantee of reducing the L2 error. In addition, our empirical evaluations show that the scaled logistic activation usually performs well.

The work of Breiman [33] restricts non-negative combination weights to prevent poor generalization errors and concludes that it is not necessary to restrict the sum of weights to equal 1. In [34], Hashem shows that linear dependence of components could be, but is not necessarily always, harmful to ensemble accuracy, whereas our work allows a mix of pre-defined and non-instantiated components as well as negative weights to provide flexibility in solution design.

Transfer Learning. In the context of one task with a very small amount of training data with another similar task that has sufficient data, transfer learning can be useful [35]. Typically the two data sets—the source and target domains—have different distributions. A neural network such as an auto-encoder is trained with source-domain data and the corresponding hidden layer weights or output labels are used for the target task. Part of transplanted weights can be kept fixed during the consequent steps, whereas others are trainable for fine-tuning [28]. This is in contrast to the composite neural network, in which the pre-trained weights are always fixed.

Model Reuse. In recent years, some proposed frameworks emphasize the reuse of fixed models [29], [30], [36]. In this framework, pre-trained models are usually connected with the main (i.e., target) model, and then the dependency is gradually weakened by removing or reducing the connections during the training process. In this way, the knowledge of the fixed model is transferred to the main model; the key point is that model reuse is different from transfer learning as well as the composite neural network.

Pre-trained models are widely applied in applications of natural language processing to improve the generation ability of the main model, such as in BERT [37] and ELMo [38]. Multi-view learning [39] is another method to improve generalization performance. In this approach, a specific task owns several sets of features corresponding to different views, just like an object

TABLE XI
COMPOSITE NETWORKS OF g_1, g_2 USING EXHAUSTIVE SEARCH, +24H

Step	Model	Training	Testing	Front-runner
3	$L(g_1^{\times}, g_2^{\times})$	5.8897	9.6059	
	$L(g_1^{\times}, g_2^{\circ})$	5.6321	9.5278	
	$L(g_1^{\circ}, g_2^{\times})$	4.9207	9.8139	
	$L(g_1^{\circ}, g_2^{\circ})$	4.5724	9.5881	
	$SL(g_1^{\times}, g_2^{\times})$	5.8941	9.6162	
	$SL(g_1^{\times}, g_2^{\circ})$	5.3703	**9.5250**	g_3
	$SL(g_1^{\circ}, g_2^{\times})$	4.7438	9.8185	
	$SL(g_1^{\circ}, g_2^{\circ})$	4.1957	9.6039	

TABLE XII
COMPOSITE NETWORKS OF g_3 AND f_5, THEN f_{W_6}, +24H

Step	Model	Training	Testing	Front-runner
4a	$L(g_3^{\times}, f_5^{\times})$	5.3349	9.3055	
	$L(g_3^{\times}, f_5^{\circ})$	5.2516	9.3186	
	$L(g_3^{\circ}, f_5^{\times})$	5.5953	9.5570	
	$L(g_3^{\circ}, f_5^{\circ})$	6.6938	9.4190	
	$SL(g_3^{\times}, f_5^{\times})$	5.5415	9.4504	
	$SL(g_3^{\times}, f_5^{\circ})$	5.1646	**9.2438**	g_{4a}
	$SL(g_3^{\circ}, f_5^{\times})$	7.2401	9.4492	
	$SL(g_3^{\circ}, f_5^{\circ})$	7.0476	9.4947	
5a	$L(g_{4a}^{\times}, f_{W_6}^{\circ})$	5.3665	9.2730	
	$L(g_{4a}^{\circ}, f_{W_6}^{\circ})$	4.3968	9.4362	
	$SL(g_{4a}^{\times}, f_{W_6}^{\circ})$	5.5986	**9.1971**	g_{5a}
	$SL(g_{4a}^{\circ}, f_{W_6}^{\circ})$	5.5421	9.4882	

observed from various perspectives, and separate models are trained accordingly. Then, the trained models for different views are combined using co-training, co-regularization, or transfer learning methods.

Air Quality Forecasting. There are several air quality prediction systems that combine different components, although these components are usually not pre-trained. In [40], Zheng et al. propose a model combining two components—an artificial neural network as the spatial classifier and a conditional random field as the temporal classifier—to infer air quality indices. Zheng et al. [41] propose a prediction model for +48h forecasting composed of four components: a temporal predictor (linear regression), a spatial predictor (neural network), a dynamic aggregator of both temporal and spatial predictors, and an inflection predictor capturing sudden changes. According to the data provided by the monitoring stations, Hsieh et al. [42] propose a system to predict the air quality class even for locations without monitoring stations. Furthermore, for locations with poor prediction, a location is recommended to install a new monitoring station for best prediction. Their inference model is based on an affinity graph. In [12], Wei et al. employ transfer learning to address the problem of big cities with a large amount of air quality data along with small cities that have insufficient data to train a model from scratch. Using pre-trained components shows strengths in flexibility in design and efficiency in training, the work in [13] presents well-thought component designs, and feature engineering and encoding that are valuable for forthcoming PM2.5 prediction studies. Yi et al. [2] propose a deep neural network consisting of a spatial transformation component and a deep distributed fusion network to fuse heterogeneous urban data to capture the factors affecting air quality. The hybird architecture of CNN and Bi-LSTM trained from scratch by Du et al. [43] is designed to learn the correlation and interdependence spatial-temporal information. In the reverse of decomposition, Qi et al. [44] integrate the three tasks, feature analysis, prediction and interpolation, into one deep learning model.

VII. CONCLUSIONS

In this work, we investigate a composite neural network composed of pre-trained components connected by differentiable activation functions. Through theoretical analysis and empirical evaluations, we show that if assumptions A1 to A4 are satisfied, especially when training data is sufficient, then a composite network has better performance than all of its components with high probability.

While the proposed theory ensures overall performance improvement, it is still not clear how to decompose a complicated problem into components and how to construct them into a composite network to yield acceptable performance. Another problem worth investigating is when the performance improvements diminish even after adding more components. Note that in real-world applications, the amount of data, the data distribution, and the data quality affect performance considerably.

REFERENCES

[1] F. Feng, X. He, X. Wang, C. Luo, Y. Liu, and T. Chua, "Temporal relational ranking for stock prediction," *ACM Trans. Inf. Syst.*, vol. 37, no. 2, pp. 27:1–27:30, 2019.

TABLE XIII
COMPOSITE NETWORKS OF g_3 AND f_{W_6}, THEN f_5, +24H

Step	Model	Training	Testing	Front-runner
4b	$L(g_3^\times, f_{W_6}^\circ)$	4.5310	9.4551	
	$L(g_3^\circ, f_{W_6}^\circ)$	4.6822	9.4677	
	$SL(g_3^\times, f_{W_6}^\circ)$	5.5339	**9.3423**	g_{4b}
	$SL(g_3^\circ, f_{W_6}^\circ)$	4.7831	9.4991	
5b	$L(g_{4b}^\times, f_5^\times)$	5.6073	9.3961	
	$L(g_{4b}^\circ, f_5^\times)$	6.6991	**9.2591**	g_{5b}
	$L(g_{4b}^\times, f_5^\circ)$	5.3298	9.2721	
	$L(g_{4b}^\circ, f_5^\circ)$	7.1666	9.5607	
	$SL(g_{4b}^\times, f_5^\times)$	5.4710	9.3313	
	$SL(g_{4b}^\times, f_5^\circ)$	5.3607	9.3130	
	$SL(g_{4b}^\circ, f_5^\times)$	6.2875	9.3586	
	$SL(g_{4b}^\circ, f_5^\circ)$	6.5281	9.5541	

TABLE XIV
SUMMARY OF ALL METHODS (RMSE)

Method	Trainable	+24h		+48h		+72h	
		Training	Testing	Training	Testing	Training	Testing
SVM	-	11.6440	10.9117	12.1246	11.5469	12.1670	11.6376
Random forests	-	3.3181	10.9386	3.4304	11.9037	3.4148	12.0917
Ensemble	1638	11.6955	11.0200	12.2609	11.3969	12.6605	11.6119
SL(Ensemble)	1638	11.5855	10.9184	12.2080	11.2815	12.5690	11.5411
DBCN$_{Relu}$	2664	12.4800	11.4540	13.3464	12.1947	14.0421	12.6546
DBCN$_{Sigm}$	4032	11.7786	10.9803	13.6521	12.4418	13.4414	12.2825
DBCN	2664	7.0560	**10.2119**	8.0678	**11.0469**	8.2305	**11.4274**
BBCN$_{Relu}$	2664	13.3711	12.4575	14.6168	13.2662	15.8200	14.0754
BBCN$_{Sigm}$	4032	12.5376	11.4600	13.0951	12.2047	13.5416	12.0388
BBCN	2664	7.1069	10.4712	7.9949	11.0935	8.4460	11.5100
Exhaustive-a	2664+α	5.1646	**9.2438**	5.0981	**10.2402**	6.7830	**10.4265**
(Include f_{W_6}), note that α =4408436, β =4449816							
Ensemble	43684	11.5253	10.7338	12.4490	11.1874	12.5822	11.4804
SL(Ensemble)	43684	11.5117	10.8125	12.3939	11.1628	12.7025	11.3376
DBCN$_{Relu}$	44710	12.9434	11.8209	14.3413	12.8331	14.3562	12.7689
DBCN$_{Sigm}$	46420	11.9444	10.9167	12.1700	**10.9474**	13.2754	11.8630
DBCN	44710	6.9705	**10.1053**	7.8941	10.9531	8.2448	**11.2541**
BBCN$_{Relu}$	44710	11.4985	10.5742	12.0386	11.0392	12.7188	11.4047
BBCN$_{Sigm}$	46420	12.4675	11.3664	13.1786	11.9285	13.3815	11.8680
BBCN	44710	6.9828	10.1938	8.5736	11.0182	9.1848	11.4153
Exhaustive-a	44710+β	5.5986	**9.1971**	5.1292	**10.2190**	7.9572	**10.3588**
Exhaustive-b	44710+β	6.6991	**9.2591**	5.6125	**10.0632**	5.7376	**10.2671**

[2] X. Yi, Z. Duan, R. Li, J. Zhang, T. Li, and Y. Zheng, "Predicting fine-grained air quality based on deep neural networks," *IEEE Transactions on Big Data*, 2020.

[3] K. Hornik, "Approximation capabilities of multilayer feedforward networks," *Neural networks*, vol. 4, no. 2, pp. 251–257, 1991.

[4] Y. Freund and R. E. Schapire, "A decision-theoretic generalization of on-line learning and an application to boosting," *Journal of computer and system sciences*, vol. 55, no. 1, pp. 119–139, 1997.

[5] T. Galanti, L. Wolf, and T. Hazan, "A theoretical framework for deep transfer learning," *Information and Inference: A Journal of the IMA*, vol. 5, no. 2, pp. 159–209, 2016.

[6] Z.-H. Zhou, *Ensemble methods: foundations and algorithms*. CRC press, 2012.

[7] Z.-H. Zhou, J. Wu, and W. Tang, "Ensembling neural networks: many could be better than all," *Artificial intelligence*, vol. 137, no. 1-2, pp. 239–263, 2002.

[8] X. Chen, S. Wang, B. Fu, M. Long, and J. Wang, "Catastrophic forgetting meets negative transfer: Batch spectral shrinkage for safe transfer learning," *32th Advances in Neural Information Processing Systems (NeurIPS)*, 2019.

[9] S. Džeroski and B. Ženko, "Is combining classifiers with stacking better than selecting the best one?" *Machine learning*, vol. 54, no. 3, pp. 255–273, 2004.

[10] M. Gashler, C. Giraud-Carrier, and T. Martinez, "Decision tree ensemble: Small heterogeneous is better than large homogeneous," in *Machine Learning and Applications, 2008. ICMLA'08. Seventh International Conference on*. IEEE, 2008, pp. 900–905.

[11] M. C. Turner, D. Krewski, W. R. Diver, C. A. Pope III, R. T. Burnett, M. Jerrett, J. D. Marshall, and S. M. Gapstur, "Ambient air pollution and cancer mortality in the cancer prevention study ii," *Environmental health perspectives*, vol. 125, no. 8, p. 087013, 2017.

[12] Y. Wei, Y. Zheng, and Q. Yang, "Transfer knowledge between cities," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 2016, pp. 1905–1914.

[13] X. Yi, J. Zhang, Z. Wang, T. Li, and Y. Zheng, "Deep distributed fusion network for air quality prediction," in *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. ACM, 2018, pp. 965–973.

[14] J. Li, H. Zhang, C.-Y. Chao, C.-H. Chien, C.-Y. Wu, C. H. Luo, L.-J. Chen, and P. Biswas, "Integrating low-cost air quality sensor networks with fixed and satellite monitoring systems to study ground-level pm2. 5," *Atmospheric Environment*, vol. 223, 2020.

[15] R. A. Horn and C. R. Johnson, *Matrix analysis*, 2nd ed. Cambridge university press, 2012.

[16] W. B. Johnson and J. Lindenstrauss, "Extensions of lipschitz mappings into a hilbert space," *Contemporary mathematics*, vol. 26, no. 189-206, p. 1, 1984.

[17] W. Rudin, *Principles of mathematical analysis*, 3rd ed. McGraw-hill New York, 1964.

[18] R. Courant and F. John, *Introduction to calculus and analysis I*. Springer Science & Business Media, 2012.

[19] M. A. Hearst, "Support vector machines," *IEEE Intelligent Systems*, vol. 13, no. 4, pp. 18–28, Jul. 1998. [Online]. Available: http://dx.doi.org/10.1109/5254.708428

[20] L. Breiman, "Random forests," *Mach. Learn.*, vol. 45, no. 1, pp. 5–32, Oct. 2001. [Online]. Available: https://doi.org/10.1023/A:1010933404324

[21] *Environmental Protection Administration*. [Online]. Available: https://opendata.epa.gov.tw/Home

[22] *Center Weather Bureau*. [Online]. Available: https://opendata.cwb.gov.tw/index

[23] D. W. Wong, L. Yuan, and S. A. Perlin, "Comparison of spatial interpolation methods for the estimation of air quality data," *Journal of Exposure Science and Environmental Epidemiology*, vol. 14, no. 5, p. 404, 2004.

[24] S. Xingjian, Z. Chen, H. Wang, D.-Y. Yeung, W.-K. Wong, and W.-c. Woo, "Convolutional lstm network: A machine learning approach for precipitation nowcasting," in *Advances in neural information processing systems*, 2015, pp. 802–810.

[25] A. Akansu and R. Haddad, *Multiresolution Signal Decomposition: Transforms, Subbands, and Wavelets*, 2nd ed. Academic Press, 2001.

[26] Y. Bengio, R. Ducharme, P. Vincent, and C. Janvin, "A neural probabilistic language model," *J. Mach. Learn. Res.*, vol. 3, pp. 1137–1155, Mar. 2003.

[27] *scikit-learn*. [Online]. Available: https://github.com/scikit-learn/scikit-learn

[28] D. Erhan, Y. Bengio, A. Courville, P.-A. Manzagol, P. Vincent, and S. Bengio, "Why does unsupervised pre-training help deep learning?" *Journal of Machine Learning Research*, vol. 11, no. Feb, pp. 625–660, 2010.

[29] Y. Yang, D.-C. Zhan, Y. Fan, Y. Jiang, and Z.-H. Zhou, "Deep learning for fixed model reuse," in *Thirty-First AAAI Conference on Artificial Intelligence*, 2017.

[30] X.-Z. Wu, S. Liu, and Z.-H. Zhou, "Heterogeneous model reuse via optimizing multiparty multiclass margin," in *International Conference on Machine Learning (ICML)*, 2019, pp. 6840–6849.

[31] D. H. Wolpert, "Stacked generalization," *Neural networks*, vol. 5, no. 2, pp. 241–259, 1992.

[32] K. M. Ting and I. H. Witten, "Issues in stacked generalization," *Journal of artificial intelligence research*, vol. 10, pp. 271–289, 1999.

[33] L. Breiman, "Stacked regressions," *Machine learning*, vol. 24, no. 1, pp. 49–64, 1996.

[34] S. Hashem, "Optimal linear combinations of neural networks," *Neural networks*, vol. 10, no. 4, pp. 599–614, 1997.

[35] S. J. Pan and Q. Yang, "A survey on transfer learning," *IEEE Transactions on knowledge and data engineering*, vol. 22, no. 10, pp. 1345–1359, 2009.

[36] J. Feng and Z.-H. Zhou, "Autoencoder by forest," in *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[37] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," *NAACL2019*, 2018.

[38] M. E. Peters, M. Neumann, M. Iyyer, M. Gardner, C. Clark, K. Lee, and L. Zettlemoyer, "Deep contextualized word representations," *NAACL2018*, 2018.

[39] J. Zhao, X. Xie, X. Xu, and S. Sun, "Multi-view learning overview: Recent progress and new challenges," *Information Fusion*, vol. 38, pp. 43–54, 2017.

[40] Y. Zheng, F. Liu, and H.-P. Hsieh, "U-air: When urban air quality inference meets big data," in *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2013, pp. 1436–1444.

[41] Y. Zheng, X. Yi, M. Li, R. Li, Z. Shan, E. Chang, and T. Li, "Forecasting fine-grained air quality based on big data," in *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 2015, pp. 2267–2276.

[42] H.-P. Hsieh, S.-D. Lin, and Y. Zheng, "Inferring air quality for station location recommendation based on urban big data," in *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 2015, pp. 437–446.

[43] S. Du, T. Li, Y. Yang, and S.-J. Horng, "Deep air quality forecasting using hybrid deep learning framework," *IEEE Transactions on Knowledge and Data Engineering*, vol. 33, no. 6, 2021.

[44] Z. Qi, T. Wang, G. Song, W. Hu, X. Li, and Z. Zhang, "Deep air learning: Interpolation, prediction, and feature analysis of fine-grained air quality," *IEEE Transactions on Knowledge and Data Engineering*, vol. 30, no. 12, 2018.

APPENDIX

A1. More details of Proofs

Proof. (of Lemma 1)

Recall that in the case of linear activate function, $g(\mathbf{x}) = L(f_1, ... f_K) = \sum_{j \in [K]^+} \theta_j f_j(\mathbf{x})$. Also recall that $\mathcal{E}_{\boldsymbol{\Theta}}(\mathbf{x}; g) = \sum_{i=1}^{N} (g(\mathbf{x}^{(i)}) - y^{(i)})^2$. To prove the existence of the minimizer, it is sufficient to find the critical point for the deferential of Eq. (1). That is, to calculate the solution, the set of equations:

$$
\nabla_{\boldsymbol{\Theta}} \mathcal{E}(\mathbf{x}; g) = \begin{bmatrix} \frac{\partial \mathcal{E}}{\partial \theta_0} \\ \vdots \\ \frac{\partial \mathcal{E}}{\partial \theta_K} \end{bmatrix} = \begin{bmatrix} 0 \\ \vdots \\ 0 \end{bmatrix},
$$

where for each $s \in [K]^+$, and

$$
\begin{aligned}
\frac{\partial \mathcal{E}}{\partial \theta_s} &= 2 \sum_{i=1}^{N} \left(g(\mathbf{x}^{(i)}) - y^{(i)} \right) \cdot f_s(\mathbf{x}^{(i)}) \\
&= 2 \sum_{i=1}^{N} \left(\sum_{j \in [K]^+} \theta_j f_j(\mathbf{x}_j^{(i)}) - y^{(i)} \right) \cdot f_s(\mathbf{x}^{(i)}) \\
&= 2 \left(\sum_{j \in [K]^+} \theta_j \langle \vec{f}_s, \vec{f}_j \rangle - \langle \vec{f}_s, \vec{y} \rangle \right).
\end{aligned}
$$

Hence, to solve $\nabla_{\boldsymbol{\Theta}} \mathcal{E}(\mathbf{x}; g) = \vec{0}$ is equivalent to solve θ_ts in the equation

$$
\left[\langle \vec{f}_s, \vec{f}_t \rangle \right]_{(K+1) \times (K+1)} \times [\theta_t]_{(K+1) \times 1} = \left[\langle \vec{f}_s, \vec{y} \rangle \right]_{(K+1) \times 1}
$$

where the indexes s, t are in $[K]^+$.

Note that linear independence of $\{\vec{f}_j\}_{j \in [K]^+}$ makes $\left[\langle \vec{f}_s, \vec{f}_t \rangle \right]_{(K+1) \times (K+1)}$ a positive-definite Gram matrix [15], which means the inversion $\left[\langle \vec{f}_s, \vec{f}_t \rangle \right]_{(K+1) \times (K+1)}^{-1}$ exists. Then the minimizer $\boldsymbol{\Theta}^*$ is solved:

$$
[\theta_t]_{(K+1) \times 1} = \left[\langle \vec{f}_s, \vec{f}_t \rangle \right]_{(K+1) \times (K+1)}^{-1} \times \left[\langle \vec{f}_s, \vec{y} \rangle \right]_{(K+1) \times 1} \tag{16}
$$

The above shows the existence of the critical points. It is easy to check that the critical point can only be the minimizer of the squared error $\mathcal{E}_{\boldsymbol{\Theta}}(\mathbf{x}; g)$. Furthermore, we immediately have $\mathcal{E}(g_{\boldsymbol{\Theta}^*}) \le \min_{j \in [K]^+} \{\mathcal{E}(f_j)\}$. $\square$

From the above proof, we can compute the minimizer for the case of the linear activation.

Corollary 1. *The closed form of the minimizer is:*

$$
\boldsymbol{\Theta}^* = [\theta_j]_{(K+1) \times 1} = \left[\langle \vec{f}_i, \vec{f}_j \rangle \right]_{(K+1) \times (K+1)}^{-1} \times \left[\langle \vec{f}_j, \vec{y} \rangle \right]_{(K+1) \times 1}.
$$

Based on Lemma 2, we can prove our next lemma

Proof. (of Lemma 3)

Apply Lemma 2 to the given $\vec{y}$ and randomly selected $\vec{f}$, then we have

$$
\Pr_{\vec{f} \in \mathbb{R}^N} \left\{ |\angle_{\vec{y}, \vec{f}} - \frac{\pi}{2}| \le \eta \right\} \ge 1 - \frac{1}{\sqrt{N}}.
$$

Also note that vectors $\vec{y}$, $\vec{f}$ and $\vec{f} - \vec{y}$ form a triangle with the three inner angles $\angle_{\vec{y}, \vec{f}}$, $\angle_{\vec{f} - \vec{y}, \vec{f}}$ and $\angle_{\vec{f} - \vec{y}, -\vec{y}}$, which means $\angle_{\vec{y}, \vec{f}} + \angle_{\vec{f} - \vec{y}, \vec{f}} + \angle_{\vec{f} - \vec{y}, -\vec{y}} = \pi$. Hence, for large N,

$$
\begin{aligned}
&\angle_{\vec{y}, \vec{f}} = \frac{\pi}{2} \Rightarrow \angle_{\vec{f} - \vec{y}, \vec{f}} \ne \frac{\pi}{2} \\
&\Rightarrow \Pr \left\{ \angle_{\vec{y}, \vec{f}} = \frac{\pi}{2} \right\} \le \Pr \left\{ \angle_{\vec{f} - \vec{y}, \vec{f}} \ne \frac{\pi}{2} \right\} \\
&\Rightarrow \Pr \left\{ \angle_{\vec{y}, \vec{f}} \approx \frac{\pi}{2} \right\} \le \Pr \left\{ \angle_{\vec{f} - \vec{y}, \vec{f}} \not\approx \frac{\pi}{2} \right\} \\
&\Rightarrow 1 - \frac{1}{\sqrt{N}} \le \Pr \left\{ \angle_{\vec{y}, \vec{f}} \approx \frac{\pi}{2} \right\} \le \Pr \left\{ \angle_{\vec{f} - \vec{y}, \vec{f}} \not\approx \frac{\pi}{2} \right\}
\end{aligned}
$$

This means there exists a small enough $\eta > 0$ s.t.

$$1 - \frac{1}{\sqrt{N}} \leq \Pr\left\{|\angle_{\vec{y},\vec{f}} - \frac{\pi}{2}| \leq \eta\right\} \leq \Pr\left\{|\angle_{\vec{f}-\vec{y},\vec{f}} - \frac{\pi}{2}| \geq \eta\right\}$$

$$\Rightarrow \frac{1}{\sqrt{N}} > \Pr\left\{|\angle_{\vec{f}-\vec{y},\vec{f}} - \frac{\pi}{2}| < \eta\right\}$$

In short, as $\angle_{\vec{f}-\vec{y},\vec{f}}$ is likely $\pi/2$, $\angle_{\vec{y},\vec{f}}$ must be less likely a vertical angle. Hence, $1 - \frac{1}{\sqrt{N}} \leq \Pr\{|\angle_{\vec{f}-\vec{y},\vec{f}} - \frac{\pi}{2}| \leq \eta\} \leq \Pr\{|\angle_{\vec{y},\vec{f}} - \frac{\pi}{2}| > \eta\}$. This completes the proof. $\square$

Proof. (of Lemma 4)

Observe that as j^* is fixed and known,

$$\Pr\left\{\nabla_{\boldsymbol{\Theta}}\mathcal{E}|_{\Theta^*=e_{\vec{j^*}}} = \vec{0}\right\}$$

$$= \Pr\left\{\langle \vec{f}_{j^*} - \vec{y}, \vec{f}_0 \rangle = 0 \wedge \cdots \wedge \langle \vec{f}_{j^*} - \vec{y}, \vec{f}_K \rangle = 0\right\}$$

$$\leq \Pr\left\{\langle \vec{f}_{j^*} - \vec{y}, \vec{f}_{j^*} \rangle = 0\right\}$$

$$< \frac{1}{\sqrt{N}}$$

The last inequality is from Lemma 3. However, in general j^* is unknown,

$$\Pr\left\{\exists \boldsymbol{\Theta}^* : \mathcal{E}(g_{\boldsymbol{\Theta}^*}) = \min_{j \in [K]^+}\{\mathcal{E}(f_j)\}\right\}$$

$$= \Pr\left\{\exists j \in [K]^+ s.t. \nabla_{\boldsymbol{\Theta}}\mathcal{E}|_{\Theta^*=e_{\vec{j}}} = \vec{0}\right\}$$

$$\leq \Pr\left\{\vee_{j=0}^{K}\left\{\langle \vec{f}_j - \vec{y}, \vec{f}_j \rangle = 0\right\}\right\}$$

$$= (K+1)\Pr\left\{\langle \vec{f} - \vec{y}, \vec{f} \rangle = 0\right\}$$

$$< \frac{K+1}{\sqrt{N}}$$

Hence,

$$\Pr\left\{\exists \boldsymbol{\Theta}^* \in \mathbb{R}^{K+1} s.t. \mathcal{E}(g_{\boldsymbol{\Theta}^*}) < \min_{j \in [K]^+}\{\mathcal{E}(f_j)\}\right\} > 1 - \frac{K+1}{\sqrt{N}}$$

$\square$

Proof. (of Lemma 7)

For Eq. (8): We first give a procedure of obtaining $g_{\boldsymbol{\Theta}_\epsilon}(\mathbf{x}^{(i)})$, then verify these settings in the procedure fit the conclusion of the first part: $\forall i \in [N], |g_{\boldsymbol{\Theta}_\epsilon}(\mathbf{x}^{(i)}) - g_{\boldsymbol{\Theta}_0^*}(\mathbf{x}^{(i)})| < \epsilon$.

Procedure for Eq. (8):

For the given ϵ and $\sigma(\cdot)$, we first find the following items based on the conclusions of Case 1 and Lemmas:

$g_{\boldsymbol{\Theta}_0^*}(\cdot)$. (By case 1)

$z_0 \in \mathbb{R}$ s.t. $\frac{d}{dz}\sigma(z) \neq 0$. (By A3)

U contains z_0. (By Lemma 5)

V contains y_0. (By Lemma 5)

$\tau : V \to U$ s.t. $\forall z \in U, \tau(\sigma(z)) = z$. (By Lemma 5)

(Denote $y_0 = \sigma(z_0)$, so $\tau(y_0) = z_0$.)
Then compute:

$$M_g = \max\left\{1, \max_{i \in [N]}\{2 \cdot |g_{\boldsymbol{\Theta}_0^*}(\mathbf{x}^{(i)})|\}\right\}$$

$$M_\sigma = \max\left\{1, \sup_{z \in U}\{2 \cdot \left(\frac{\sigma(z) - \sigma(z_0)}{z - z_0}\right)^2\}\right\}$$

$$M_\tau = \max\left\{1, \sup_{z \in U}\{|\tau^{(2)}(\sigma(z) - \sigma(z_0))|\}\right\}$$

$$M_\gamma = \lceil \log_2(M_g M_\sigma M_\tau \epsilon^{-1})\rceil + 1$$

$$\gamma_0 = \sup_{z \in U}\left\{r = |z - z_0| : (z_0 - r, z_0 + r) \subset U\right\}$$

$$\gamma = \min\left\{\gamma_0, 2^{-M_\gamma}\right\}$$

$$M_0 = \gamma^{-1} M_g$$

$$M_1 = M_\sigma M_\tau$$

Define:

$$L_{(0),\epsilon}(\mathbf{x}) = M_0^{-1} g_{\boldsymbol{\Theta}_0^*}(\mathbf{x}) + z_0$$

$$L_{(1),\epsilon}(y) = M_0 \cdot \tau^{(1)}(y_0) \cdot y + M_0 \cdot \left(z_0 - \tau^{(1)}(y_0) \cdot y_0\right)$$

Verification:
First observe that $L_{(0),\epsilon}(\mathbf{x})$ is a linear combination with a bias, i.e., an affine mapping, since $g_{\boldsymbol{\Theta}_0^*}(\mathbf{x})$ itself is an affine mapping. Similarly, $L_{(1),\epsilon}(y)$ is an affine mapping of y.

Next, for all $i \in [N]$, $L_{\boldsymbol{\Theta}_{0,\epsilon}}(\mathbf{x}^{(i)}) = M_0^{-1} g_{\boldsymbol{\Theta}_0^*}(\mathbf{x}^{(i)}) + z_0 \in (-\gamma + z_0, z_0 + \gamma) \subset U$ since $\gamma \leq \frac{\gamma_0}{2}$ and $(-\frac{\gamma_0}{2} + z_0, z_0 + \frac{\gamma_0}{2}) \subset U$.
Hence, by Lemma 5,

$$\tau\left(\sigma\left(L_{\boldsymbol{\Theta}_{0,\epsilon}}(\mathbf{x}^{(i)})\right)\right) = L_{\boldsymbol{\Theta}_{0,\epsilon}}(\mathbf{x}^{(i)}).$$

Now let $z \in (-\gamma + z_0, z_0 + \gamma)$ and $y = \sigma(z)$, then by Lemma 6 and Eq. (7),

$$|\tau(y) - \left(\tau(y_0) + \tau^{(1)}(y_0)(y - y_0)\right)|$$

$$= \frac{\tau^{(2)}(c(y - y_0))}{2!}(y - y_0)^2$$

$$< 2 \cdot \sup_{z \in U}\left\{|\tau^{(2)}(\sigma(z) - \sigma(z_0))| \cdot \left(\frac{\sigma(z) - \sigma(z_0)}{z - z_0}\right)^2\right\} \cdot (z - z_0)^2$$

$$\leq M_\tau M_\sigma \gamma^2 = M_1 \gamma^2$$

Replace y with $\sigma(z)$ and simplify the expression in the absolute value symbol, then we have $\tau(y) = \tau(\sigma(z)) = z$. Furthermore, $\tau(y_0) + \tau^{(1)}(y_0)(y - y_0) = \tau^{(1)}(y_0) \cdot y + \left(\tau(y_0) - \tau^{(1)}(y_0) \cdot y_0\right)$. Then replace z with $L_{\boldsymbol{\Theta}_{0,\epsilon}}(\mathbf{x}^{(i)})$, and $\tau(y_0)$ with z_0,

$$|M_0^{-1} g_{\boldsymbol{\Theta}_0^*}(\mathbf{x}) + z_0 - \left\{z_0 + \tau^{(1)}(y_0)\left(\sigma\left(L_{\boldsymbol{\Theta}_{0,\epsilon}}(\mathbf{x}^{(i)})\right) - y_0\right)\right\}|$$

$$< M_1 \gamma^2$$

This means that

$$|g_{\boldsymbol{\Theta}_0^*}(\mathbf{x}) - L_{\boldsymbol{\Theta}_{1,\epsilon}}\left(\sigma\left(L_{\boldsymbol{\Theta}_{0,\epsilon}}(\mathbf{x}^{(i)})\right)\right)| < M_0 M_1 \gamma^2$$

$$\Rightarrow |g_{\boldsymbol{\Theta}_0^*}(\mathbf{x}) - g_{\boldsymbol{\Theta}_\epsilon}(\mathbf{x})| < M_0 M_1 \gamma^2$$

Recall that $\gamma \leq 2^{-M_\gamma} < \frac{\epsilon}{M_g M_\sigma M_\tau}$. Hence,

$$M_0 M_1 \gamma^2 = \gamma^{-1} M_g M_\sigma M_\tau \gamma^2 = M_g M_\sigma M_\tau \gamma < \epsilon$$

achieve the goal of the first part of this Lemma.

For Eq. (9): For the second part, we claim the following settings satisfy $\mathcal{E}(g_{\boldsymbol{\Theta}_\epsilon}) \leq \frac{2\mathcal{E}(g_{\boldsymbol{\Theta}_0^*}) + \mathcal{E}(f_{j^*})}{3} < \mathcal{E}(f_{j^*})$.
Procedure for Eq. (9):
Compute and then set these:

$$M_2 = \max_{i \in [N]}\left\{|g_{\boldsymbol{\Theta}_0^*}(\mathbf{x}^{(i)}) - y^{(i)}|\right\}$$

$$\epsilon = \frac{\mathcal{E}(f_{j^*}) - \mathcal{E}(g_{\boldsymbol{\Theta}_0^*})}{4N(2M_2 + 1)}$$

Verification:

Observe that

$$\mathcal{E}(g_{\mathbf{\Theta}_0^*}) < \mathcal{E}(f_{j^*}) \Rightarrow$$

$$\max_{i \in [N]} \left\{ (f_{j^*}(\mathbf{x}^{(i)}) - y^{(i)})^2 - (g_{\mathbf{\Theta}_0^*}(\mathbf{x}^{(i)}) - y^{(i)})^2 \right\} > 0$$

$$\mathcal{E}(g_{\mathbf{\Theta}_0^*}) + \frac{\mathcal{E}(f_{j^*}) - \mathcal{E}(g_{\mathbf{\Theta}_0^*})}{3} = \frac{2\mathcal{E}(g_{\mathbf{\Theta}_0^*}) + \mathcal{E}(f_{j^*})}{3} < \mathcal{E}(f_{j^*})$$

Besides,

$$N \cdot (2M_2 + 1) \cdot \epsilon < \frac{\mathcal{E}(f_{j^*}) - \mathcal{E}(g_{\mathbf{\Theta}_0^*})}{3}$$

and

$$|g_{\mathbf{\Theta}_\epsilon}(\mathbf{x}) - g_{\mathbf{\Theta}_0^*}(\mathbf{x})| < \epsilon$$
$$\Rightarrow |(g_{\mathbf{\Theta}_\epsilon}(\mathbf{x}) - y) - (g_{\mathbf{\Theta}_0^*}(\mathbf{x}) - y)| < \epsilon$$
$$\Rightarrow 0 \le |g_{\mathbf{\Theta}_\epsilon}(\mathbf{x}) - y| < |g_{\mathbf{\Theta}_0^*}(\mathbf{x}) - y| + \epsilon$$
$$\Rightarrow (g_{\mathbf{\Theta}_\epsilon}(\mathbf{x}) - y)^2 < (|g_{\mathbf{\Theta}_0^*}(\mathbf{x}) - y| + \epsilon)^2$$

Hence, based on above observations we have

$$\mathcal{E}(g_{\mathbf{\Theta}_\epsilon}) = \sum_{i \in [N]} (g_{\mathbf{\Theta}_\epsilon}(\mathbf{x}^{(i)}) - y^{(i)})^2$$

$$< \sum_{i \in [N]} \{|g_{\mathbf{\Theta}_0^*}(\mathbf{x}^{(i)}) - y^{(i)}| + \epsilon\}^2$$

$$= \sum_{i \in [N]} (g_{\mathbf{\Theta}_0^*}(\mathbf{x}^{(i)}) - y^{(i)})^2$$

$$\quad + \sum_{i \in [N]} \left\{ 2\epsilon \cdot |g_{\mathbf{\Theta}_0^*}(\mathbf{x}^{(i)}) - y^{(i)}| + \epsilon^2 \right\}$$

$$= \mathcal{E}(g_{\mathbf{\Theta}_0^*}) + \epsilon \cdot \sum_{i \in [N]} \left(2|g_{\mathbf{\Theta}_0^*}(\mathbf{x}^{(i)}) - y^{(i)}| + \epsilon \right)$$

$$\le \mathcal{E}(g_{\mathbf{\Theta}_0^*}) + \epsilon \cdot N \cdot (2M_2 + 1)$$

$$< \mathcal{E}(g_{\mathbf{\Theta}_0^*}) + \frac{\mathcal{E}(f_{j^*}) - \mathcal{E}(g_{\mathbf{\Theta}_0^*})}{3}$$

$$= \frac{\mathcal{E}(f_{j^*}) + 2\mathcal{E}(g_{\mathbf{\Theta}_0^*})}{3}$$

$$< \mathcal{E}(f_{j^*})$$

which means that $\mathcal{E}(g_{\mathbf{\Theta}_\epsilon}) < \min_{j \in [K]^+} \{\mathcal{E}(f_j)\}$. The proof is complete. $\qquad \square$

Proof. (of Proposition 1)

Let

$$D(\alpha_0, \alpha_1)$$
$$= \sum_{i \in [N]} (f_1(\mathbf{x}^{(i)}) - y^{(i)})^2 - \left(\alpha_0 f_0(\mathbf{x}^{(i)}) + \alpha_1 f_1(\mathbf{x}^{(i)}) - y^{(i)} \right)^2.$$

First observe that $D(0,1) = 0$ and hence if $\nabla D(0,1) \ne (0,0)$ then it is easy to know that $\exists (\alpha_0^*, \alpha_1^*)$ s.t. $D(\alpha_0^*, \alpha_1^*) > 0$.

$$\nabla D(\alpha_0, \alpha_1) = -2 \cdot \begin{bmatrix} \langle \alpha_0 \vec{f_0} + \alpha_1 \vec{f_1} - \vec{y}, \vec{f_0} \rangle \\ \langle \alpha_0 \vec{f_0} + \alpha_1 \vec{f_1} - \vec{y}, \vec{f_1} \rangle \end{bmatrix}$$

Then, by considering $(\alpha_0, \alpha_1) = (0,1)$ we have

$$\nabla D(0,1) = -2 \cdot \begin{bmatrix} \langle \vec{f_1} - \vec{y}, \vec{f_0} \rangle \\ \langle \vec{f_1} - \vec{y}, \vec{f_1} \rangle \end{bmatrix}$$

Apply Lemma 3,

$$\Pr\left\{ \nabla D|_{\Theta^* = e_{\vec{j^*}}} = \vec{0} \right\}$$
$$\le \Pr\{\exists j \in [1]^+ s.t. \langle \vec{f_j} - \vec{y}, \vec{f_j} \rangle = 0\}$$
$$< \frac{2}{\sqrt{N}}$$

That is,

$$\begin{aligned}
&\Pr\{\exists(\alpha_0, \alpha_1) s.t. D(\alpha_0, \alpha_1) > 0\} \\
&\geq \Pr\{\nabla D(0, 1) \neq \vec{0}\} \\
&> 1 - \frac{2}{\sqrt{N}}
\end{aligned}$$

$\square$

Proof. (of Lemma 8)

We first prove this lemma of linear activation, and then similar to previous section apply Lemma 7 to address the non-linear activation. For the linear activation, it can be proved by induction.

Base case: It is done in Proposition 1.

Inductive step: Suppose as $J = k - 1$ the statement is true. That is, $g_{k-1} = L_\Theta(f_1, ..., f_{k-1})$ and with probability at least $1 - \frac{K}{\sqrt{N}}$, there is Θ s.t. $\mathcal{E}(g_{K-2}) > \mathcal{E}_\Theta(g_{K-1})$. As $J = k$, let f_0 and f_1 in Proposition 1 be g_{k-1} and f_k respectively. Then we have $\alpha_0 g_{k-1} + \alpha_1 f_k$ as the composite network. Repeat the argument in the previous proposition, then we can conclude with probability at least $1 - \frac{k+1}{\sqrt{N}}$ there is (α_0, α_1) s.t. $\mathcal{E}(g_{K-1}) > \mathcal{E}_\Theta(\alpha_0 g_{k-1} + \alpha_1 f_k)$. Note that $\alpha_0 g_{k-1} + \alpha_1 f_k$ is a possible form of g_K. So the statement holds. The details are as follows:

$$\begin{aligned}
&D(\alpha_0, \alpha_1) \\
&= \sum_{i \in [N]} (g_{k-1}(\mathbf{x}^{(i)}) - y^{(i)})^2 - \left(\alpha_0 g_{k-1}(\mathbf{x}^{(i)}) + \alpha_1 f_k(\mathbf{x}^{(i)}) - y^{(i)}\right)^2.
\end{aligned}$$

First observe that $D(1, 0) = 0$ and hence if $\nabla D(1, 0) \neq \vec{0}$ then it is easy to know that $\exists(\alpha_0^*, \alpha_1^*)$ s.t. $D(\alpha_0^*, \alpha_1^*) > 0$.

$$\nabla D(\alpha_0, \alpha_1) = -2 \cdot \begin{bmatrix} \langle \alpha_0 \vec{g}_{k-1} + \alpha_1 \vec{f_k} - \vec{y}, \vec{g}_{k-1} \rangle \\ \langle \alpha_0 \vec{g}_{k-1} + \alpha_1 \vec{f_k} - \vec{y}, \vec{f_k} \rangle \end{bmatrix}$$

Then,

$$\nabla D(1, 0) = -2 \cdot \begin{bmatrix} \langle \vec{g}_{k-1} - \vec{y}, \vec{g}_{k-1} \rangle \\ \langle \vec{g}_{k-1} - \vec{y}, \vec{f_k} \rangle) \end{bmatrix}$$

Apply Lemma 4 and by Induction hypothesis, we have

$$\begin{aligned}
&\Pr\left\{\nabla D|_{\Theta^* = \vec{e_{j*}}} = \vec{0}\right\} \\
&\leq \Pr\{\langle \vec{g}_{k-1} - \vec{y}, \vec{g}_{k-1} \rangle = 0\} + \Pr\{\langle \vec{f_k} - \vec{y}, \vec{f_k} \rangle = 0\} \\
&< \frac{k}{\sqrt{N}} + \frac{1}{\sqrt{N}} = \frac{k+1}{\sqrt{N}}
\end{aligned}$$

Thus,

$$\begin{aligned}
&\Pr\{\exists(\alpha_0, \alpha_1) s.t. D(\alpha_0, \alpha_1) > 0\} \\
&\geq \Pr\left\{\nabla D|_{\Theta^* = \vec{e_{j*}}} \neq \vec{0}\right\} \\
&> 1 - \frac{k+1}{\sqrt{N}}
\end{aligned}$$

This completes the inductive step.

For the non-linear activation, repeat the argument of Lemma 7 to obtain a proper g_{Θ_ϵ} corresponding to the given ϵ and the linear mapping $g_{\Theta_0^*}$, and a small enough ϵ can yield a proper Θ_ϵ that fits the conclusion of $\mathcal{E}(g_{K-1}) > \mathcal{E}_{\Theta_\epsilon}(g_K)$. The probability of existence is inherently obtained as the same as in Lemma 7. $\square$

Proof. (of Lemma 9)

Observe that for the given set of pre-trained components $\{f_j\}_{j \in [K]}$ and by the definition of g_{K-1}, f_K is not a component of g_{K-1}. Hence, if the activation functions used in the construction of g_{K-1} are all linear, the assumption A1 implies that $\vec{g}_{K-1}$ is linear independent of $\vec{f}_K$. Furthermore, if there is at least one non-linear activation function used in the construction of g_{K-1}, then as N is large enough, Lemma 2 implies that $\vec{g}_{K-1}$ and $\vec{f}_K$ are not parallel with a very high probability. This means the assumption that $\vec{g}_{K-1}$ is linear independent of $\vec{f}_K$ is reasonable. Furthermore, this implies that the events $E_1 : \exists \Theta s.t. \mathcal{E}_\Theta(g_K) < min\{\mathcal{E}(g_{k-1}), \mathcal{E}(f_k)\}$, and $E_2 : \mathcal{E}(g_{K-1}) < \cdots < \min_{j \in [K]^+}\{\mathcal{E}(f_j)\}$, are independent. Hence, $\Pr\{E_1 | E_2\} = \Pr\{E_1\}$. $\square$

Proof. (of Theorem 2)

For a set of given K pre-trained components, $g_k := L_{(k)}(\sigma(L_{(k-1)}(\cdots L_{(1)}(\sigma(L_{(0)}(f_1, \cdots, f_K)))\cdots)))$ is one of possible H-hidden layer composite network. Hence obviously,

$$
\Pr\left\{\exists \boldsymbol{\Theta}^* : \mathcal{E}(g_{\boldsymbol{\Theta}^*}) < \min_{j\in[K]^+}\{\mathcal{E}(f_j)\}\right\}
$$

$$
\geq \Pr\left\{\mathcal{E}(g_H) < \mathcal{E}(g_{H-1}) < \cdots < \mathcal{E}(g_1) < \min_{j\in[K]^+}\{\mathcal{E}(f_j)\}\right\}
$$

$$
\geq \Pr\left\{\mathcal{E}(g_1) < \min_{j\in[K]^+}\{\mathcal{E}(f_j)\}\right\}
$$

$$
\times \Pr\left\{\mathcal{E}(g_2) < \mathcal{E}(g_1) \mid \mathcal{E}(g_1) < \min_{j\in[K]^+}\{\mathcal{E}(f_j)\}\right\} \times \cdots \times
$$

$$
\Pr\left\{\mathcal{E}(g_H) < \mathcal{E}(g_{H-1}) \mid \mathcal{E}(g_{H-1}) < \cdots < \min_{j\in[K]^+}\{\mathcal{E}(f_j)\}\right\}
$$

$$
= \left(1 - \frac{K+1}{\sqrt{N}}\right)^H
$$

The last inequality is based on Lemmas 8 and 9:

$$
\min_{k\in[H]}\{P_k\} \geq 1 - \frac{K+1}{\sqrt{N}},
$$

where

$$
P_k = \Pr\left\{\exists \boldsymbol{\Theta} : \mathcal{E}(g_k) < \mathcal{E}(g_{k-1}) \mid \mathcal{E}(g_{k-1}) < \cdots < \min_{j\in[K]^+}\{\mathcal{E}(f_j)\}\right\}
$$

$$
= \Pr\{\exists \boldsymbol{\Theta} : \mathcal{E}(g_k) < \mathcal{E}(g_{k-1})\} \text{ by Lemma 9.}
$$

This completes the proof. □

22

A2. More Details of Experiments

In Table XV shows the details of each component:

TABLE XV
ARCHITECTURES AND HYPERPARAMETERS OF COMPONENTS

Comp.	Descriptions
	Descriptions: attention (decoder) layer (Att), time length $l \in \{24, 48, 72\}$, batch normalization (BN), convolutional layer (Cvl), max-pooling (MaxP), flatten layer (Fltn), dense layer (Den), dropout (Drop)
f_1	Input layer: $(30 \times 38) \times 9$ hidden layers: BN, Cvl-1$(30 \times 38$, 32 filters), MaxP, Cvl-2$(15 \times 19$, 16 filters), MaxP, Fltn, LSTM(150, time l), Att, Fltn, Den, Drop(0.2), Den Output layer: Den(RELU), 18 total parameters: 917,510
f_2	Input layer: $(30 \times 38) \times 4$ hidden layers: same with f_1 Output layer: Den(RELU), 18 total parameters: 916,918
f_3	Input layer: $(30 \times 38) \times 9$ hidden layers: Fltn, BN, Den-1(100), Den-2(100) Output layer: Den(RELU), 18 total parameters: 1,038,054
f_4	Input layer: $(30 \times 38) \times 4$ hidden layers: same with f_3 Output layer: Den(RELU), 18 total parameters: 582,038
f_5	Input layer: one-hot $(24 + 7 + 12)$ hidden layers: LSTM(150, time l), Att, Fltn, Den, Drop(0.2), Den Output layer: Den(Scaled-Logistic), 18 total parameters: 953,916
f_{W_6}	Input layer: $(30 \times 38) \times 4$ hidden layers: Cvl-1$(30 \times 38$, 16 filters) Cvl-2$(15 \times 19$, 16 filters), MaxP, Fltn, Output layer: Den(RELU), 18 total parameters: 41,380

TABLE XVI
COMPOSITE NETWORKS BY ALGO 1: DBCN, NEXT 48HR.

Model	RMSE		parameter	note
	Training	**Testing**	trainable/total	
$g_1 \leftarrow f_1$			0/	
$L(g_1, f_5)$	8.1850	**11.0995**	666/1872092	g_2
$SL(g_1, f_5)$	8.1675	11.2399	666/1872092	
$L(g_2, f_4)$	8.1520	11.2632	666/2454796	
$SL(g_2, f_4)$	8.1902	**11.1647**	666/2454796	g_3
$L(g_3, f_3)$	8.1382	**11.1163**	666/3493516	g_4
$SL(g_3, f_3)$	8.1085	11.1442	666/3493516	
$L(g_4, f_2)$	8.0361	11.0991	666/4411100	
$SL(g_4, f_2)$	8.0678	**11.0469**	666/4411100	g_5
$L(g_5, f_{W_6})$	7.8941	**10.9531**	42046/4453146	g_6
$SL(g_5, f_{W_6})$	7.9009	10.9754	42046/4453146	

TABLE XVII
COMPOSITE NETWORKS BY ALGO 1: DBCN, NEXT 72HR.

| Model | RMSE | | parameter | |
	Training	Testing	trainable/total	note
$g_1 \leftarrow f_5$				
$L(g_1, f_1)$	8.4308	11.3572	666/1872092	
$SL(g_1, f_1)$	8.2979	**11.3323**	666/1872092	g_2
$L(g_2, f_2)$	8.3579	**11.3634**	666/2789676	g_3
$SL(g_2, f_2)$	8.3252	11.4001	666/2789676	
$L(g_3, f_4)$	8.4116	**11.4195**	666/3372380	g_4
$SL(g_3, f_4)$	8.6230	11.4530	666/3372380	
$L(g_4, f_3)$	8.2305	**11.4274**	666/4411100	g_5
$SL(g_4, f_3)$	8.1284	11.4482	666/4411100	
$L(g_5, f_{W_6})$	8.2448	**11.2541**	42046/4453146	g_6
$SL(g_5, f_{W_6})$	8.2125	11.3232	42046/4453146	

TABLE XVIII
COMPOSITE NETWORKS BY ALGO 2: BBCN, NEXT 48HR.

| Model | RMSE | | parameter | |
	Training	Testing	trainable/total	note
$h_{0,1} \leftarrow f_1, h_{0,2} \leftarrow f_2$				
$L(h_{0,1}, h_{0,2})$	6.1001	11.1004	666/1835094	
$SL(h_{0,1}, h_{0,2})$	5.5894	**11.0907**	666/1835094	$h_{1,1}$
$h_{0,3} \leftarrow f_3, h_{0,4} \leftarrow f_4$				
$L(h_{0,3}, h_{0,4})$	11.8311	**11.5098**	666/1620758	$h_{1,2}$
$SL(h_{0,3}, h_{0,4})$	11.6587	11.5436	666/1620758	
$L(h_{1,1}, h_{1,2})$	8.2277	11.1739	666/3456518	
$SL(h_{1,1}, h_{1,2})$	7.9990	**11.0935**	666/3456518	$h_{2,1}$
$h_{2,2} \leftarrow h_{1,3} \leftarrow h_{0,5} \leftarrow f_5$				
$L(h_{2,1}, h_{2,2})$	8.0273	11.0808	666/4411100	
$SL(h_{2,1}, h_{2,2})$	7.9949	**11.0516**	666/4411100	$h_{3,1}$
$g_5 \leftarrow h_{3,1}$				
$L(g_5, f_{W_6})$	8.5736	**11.0182**	42046/4453146	g_6
$SL(g_5, f_{W_6})$	7.7346	11.0208	42046/4453146	

TABLE XIX
COMPOSITE NETWORKS BY ALGO 2: BBCN, NEXT 72HR.

| Model | RMSE | | parameter | |
	Training	Testing	trainable/total	note
$h_{0,1} \leftarrow f_1, h_{0,2} \leftarrow f_2$				
$L(h_{0,1}, h_{0,2})$	7.7873	11.4480	666/1835094	
$SL(h_{0,1}, h_{0,2})$	7.9830	**11.4198**	666/1835094	$h_{1,1}$
$h_{0,3} \leftarrow f_3, h_{0,4} \leftarrow f_4$				
$L(h_{0,3}, h_{0,4})$	12.0284	**11.7405**	666/1620758	$h_{1,2}$
$SL(h_{0,3}, h_{0,4})$	12.0460	11.8005	666/1620758	
$L(h_{1,1}, h_{1,2})$	8.3884	11.7030	666/3456518	
$SL(h_{1,1}, h_{1,2})$	8.5149	**11.5703**	666/3456518	$h_{2,1}$
$h_{2,2} \leftarrow h_{1,3} \leftarrow h_{0,5} \leftarrow f_5$				
$L(h_{2,1}, h_{2,2})$	8.4460	**11.5100**	666/4411100	$h_{3,1}$
$SL(h_{2,1}, h_{2,2})$	8.4093	11.5526	666/4411100	
$g_5 \leftarrow h_{3,1}$				
$L(g_5, f_{W_6})$	9.1848	**11.4153**	42046/4453146	g_6
$SL(g_5, f_{W_6})$	9.0706	11.4474	42046/4453146	

TABLE XX
SUMMARY OF ALL METHODS (MAE)

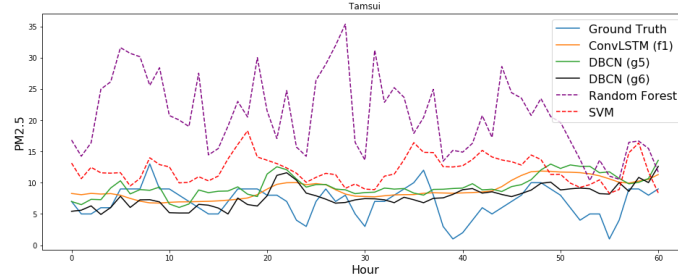| Method	+24h Training	+24h Testing	+48h Training	+48h Testing	+72h Training	+72h Testing
SVM	8.0701	**7.7026**	8.5887	**8.3525**	8.6831	**8.6157**
Random forests	2.3956	8.3523	2.5007	9.2156	2.5131	9.4219
Ensemble	8.8470	8.5245	9.1113	8.7430	9.3899	8.8767
SL(Ensemble)	8.7689	8.4413	9.0365	8.6116	9.3121	8.8657
$DBCN_{Relu}$	9.3082	8.7309	9.8383	9.1429	10.4367	9.6356
$DBCN_{Sigm}$	8.6927	8.2972	10.0452	9.6134	9.9240	9.5408
DBCN	3.7188	**7.7868**	4.3161	**8.4258**	4.3261	**8.8250**
$BBCN_{Relu}$	9.7099	9.2274	10.6558	10.0856	11.6371	10.5833
$BBCN_{Sigm}$	9.1872	8.7346	9.7944	9.5822	9.6479	8.8532
BBCN	3.7676	7.9866	4.4403	8.5564	4.5324	8.8781
Exhaustive-a	2.8646	**6.8319**	2.8078	**7.6136**	3.6612	**7.7701**
(Include f_{W_6})						
Ensemble	8.7257	8.2739	8.9901	**8.2476**	9.2801	8.7263
SL(Ensemble)	8.7470	8.3988	9.0098	8.3065	9.2281	**8.4536**
$DBCN_{Relu}$	9.3969	8.5961	10.2851	9.5191	10.4696	9.4412
$DBCN_{Sigm}$	8.7275	8.1512	8.9281	8.2625	9.6004	8.8777
DBCN	3.6608	**7.5614**	4.2419	8.2766	4.4143	8.5776
$BBCN_{Relu}$	8.5198	8.0122	8.9259	8.4752	9.2290	8.5274
$BBCN_{Sigm}$	9.1376	8.6430	9.6253	9.0825	9.6115	8.8030
BBCN	3.6698	7.6156	4.6652	8.4528	4.8884	8.6097
Exhaustive-a	3.1250	**6.7757**	2.8133	**7.5986**	4.5569	**7.6798**
Exhaustive-b	3.8088	**6.9032**	3.1118	**7.5156**	3.0740	**7.6561**

Fig. 6. Case study for Tamsui station

Figure 6 illustrates a typical example of the next-24-hour predictions of various models, including g_5 and g_6 of DBCN, ConvLSTM, SVM and random forest, of Tamsui for 60 hours starting from 9:00 pm, October 22, 2016. f_1 is a ConvLSTM model that its prediction is central to the average of the ground truth. DBCN (g_6) considers the one extra weather feature, i.e., the chance of rain in the future, that it produces a lower PM2.5 prediction than DBCN(g_5). In this duration, the traditional machine learning methods SVM and RF usually overestimated, although they apply the same weather and pollutant features.