Exploring Potential Differences Among Various Model Architectures in Heterogeneous Time Series Through Search Methods

Chen Yang
School of Computer Science and Engineering
Beihang University
Beijing, China
stx221166@buaa.edu.cn

***Abstract*—Many tasks require execution under conditions where dataset information is partially missing. For example, in unsupervised learning and domain adaptation, datasets are often provided without labels due to the high cost of human annotation. We present another task with incomplete dataset information. In deep learning, tuning the architecture's hyperparameters is crucial for model performance but can be time-consuming. We address this by adjusting hyperparameters under an uncertain dataset condition: When label acquisition is delayed, necessitating hyperparameter adjustments without full dataset information. Different hyperparameters are typically suited to different data, a fact driven by the heterogeneities within the known datasets. Dataset uncertainty adds another layer of complexity to this issue. To deal with the uncertainty, we propose several loss functions aimed at identifying probable datasets in the absence of complete information. Our focus is particularly on the application of this method to time series data. Experiments conducted on nine real-world time series datasets illustrate the effectiveness of our proposed method.**

***Index Terms*—Deep Learning, Time Series.**

I. Introduction

In deep learning, regression data are typically represented as pairs $(\boldsymbol{X}, \boldsymbol{Y})$, where $\boldsymbol{X}$ denotes the input and $\boldsymbol{Y}$ the associated label. However, different aspects of the data can present varying levels of challenge. For example, in unsupervised learning [1] and domain adaptation [2], labels are often difficult to obtain because they require human annotation, which introduces uncertainty into the dataset information.

Another scenario involves situations where the input samples $\boldsymbol{X}$ are available well before the labels $\boldsymbol{Y}$, due to the time-consuming nature of human annotation or because the task target for $\boldsymbol{Y}$ is not yet defined at the time of collecting inputs. In such cases, it is assumed that computing resources are plentiful prior to acquiring labels. Given the critical role of architecture hyperparameters in model performance, a pertinent question arises: can these parameters be compared and adjusted using only the input sample information $\boldsymbol{X}$? Selecting hyperparameters based on $\boldsymbol{X}$ alone could save time when testing various configurations once $\boldsymbol{Y}$ becomes available. Moreover, since $\boldsymbol{X}$ generally requires more storage space than $\boldsymbol{Y}$, it likely contains more information, suggesting that predicting architecture comparisons using only $\boldsymbol{X}$ is plausible.

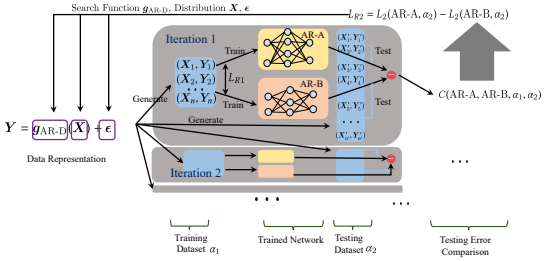

Fig. 1. Probable dataset searching procedure, where models are trained in α_1 and tested in α_2. The testing difference of two architectures are used to search probable dataset.

To address the challenges posed by uncertain dataset information, we propose a method for searching probable datasets to predict architecture comparisons. Inspired by recent neural network convergence studies [3], [4], [5], [6], [7], our approach searches for probable datasets using provided input sample information $\boldsymbol{X}$. Specifically, the comparison between two sets of hyperparameters can be predicted by identifying probable datasets where one architecture outperforms another. We use a neural network to approximate the dataset's regression function and apply various loss functions to find probable datasets where a trained architecture excels over another during testing.

Our method assumes that the compared architectures should perform competitively on the searched datasets. Empirically, we select architectures known to perform well on similar data domains from previous experience, focusing on datasets where at least one of the compared architectures can perform adequately. The neural network used to approximate the regression function mirrors the architecture being evaluated.

II. Method

A. Dataset Representation

When a sample in a regression dataset is denoted by a random variable vector with the form $(\boldsymbol{X}, \boldsymbol{Y})$, the relationship between $\boldsymbol{X}$ and $\boldsymbol{Y}$ can be represented as a regression function $\boldsymbol{f}_0(\boldsymbol{x}) = \mathbf{E}\{\boldsymbol{Y}|\boldsymbol{X} = \boldsymbol{x}\}$, where $\mathbf{E}$ is the expectation. In recent deep neural network convergence theorem [3], [4], [5], [6], [7], it is proved that $\boldsymbol{f}_0$ could be converged by multi-layer

979-8-3315-2096-0/25/$31.00 ©2025 IEEE

fully connected neural networks trained with enough samples, where $\boldsymbol{f}_0$ satisfies some requirements and assumptions. Here, we assume that the convergence theorem is also true with other types of deep neural networks. We then use a neural network to approximate a regression function with two reasons: 1, as a function, neural network also satisfies the requirements and assumptions of a regression function. 2, there is always a neural network $\boldsymbol{g}$ that the difference between $\boldsymbol{f}_0$ and $\boldsymbol{g}$ is smaller than a given positive value because of the definition of convergence.

As a result, a regression dataset can be expressed by

$$\boldsymbol{Y} = \boldsymbol{g}(\boldsymbol{X}) + \boldsymbol{\epsilon}, \tag{1}$$

where the neural network $\boldsymbol{g}$ has a small enough similarity to $\boldsymbol{f_0}$, $\boldsymbol{\epsilon}$ is a random disturbance of regression function with 0 expectation. Then, there are three components influencing the regression dataset, the regression function $\boldsymbol{g}$, $\boldsymbol{X}$ distribution and random disturbance distribution $\boldsymbol{\epsilon}$. $\boldsymbol{\epsilon}$ should be much smaller than $\boldsymbol{g}$ in the dataset so the relation between inputs and outputs could be easily distinguished from random disturbance.

There is another additional assumption of the regression dataset: The regression function should not be too complex that no compared architecture can perform well in the comparison. It also assumes that the compared architectures in the searching method are selected with empirical knowledge to provide competitive performance in an uncertain dataset with enough training samples. Practically, the assumption is satisfied by setting the regression function $\boldsymbol{g}$ with the same architecture as a compared architecture.

Formally, to train a neural network $\boldsymbol{g}_{\text{AR-A}}$ with its parameters $\boldsymbol{\theta}_{\text{AR-A}}$ with a training dataset α_1, error value Err is introduced as a smaller the better value to evaluate the performance of a model in the dataset α_1. Then training $\boldsymbol{g}_{\text{AR-A}}$ with a training dataset α_1 aims to find parameters $\boldsymbol{\theta}^*_{\text{AR-A}}(\alpha_1)$ that

$$\boldsymbol{\theta}^*_{\text{AR-A}}(\alpha_1) = \arg\min_{\boldsymbol{\theta}_{\text{AR-A}}} \text{Err}(\alpha_1, \boldsymbol{g}_{\text{AR-A}}(\cdot, \boldsymbol{\theta}_{\text{AR-A}})), \tag{2}$$

where smaller error value means better model performance in training dataset α_1. In regression task, Root Mean Square Error (RMSE) is a widely used error value: $\text{RMSE} = \sqrt{\frac{1}{K}\|\hat{\boldsymbol{y}} - \boldsymbol{y}\|_2^2}$, where $\|\cdot\|_2$ is the l-2 norm, $\boldsymbol{y}$ is the true value, $\hat{\boldsymbol{y}}$ is the model output and K is the dimension of $\boldsymbol{y}$.

The difference of the error value between two architectures on a testing dataset α_2 can be used to compare the performance of two architectures AR-A and AR-B:

$$\begin{aligned} C(\text{AR-A}, \text{AR-B}, \alpha_1, \alpha_2) = \text{Err}(\alpha_2, \boldsymbol{g}_{\text{AR-A}}(\cdot, \boldsymbol{\theta}^*_{\text{AR-A}}(\alpha_1))) \\ - \text{Err}(\alpha_2, \boldsymbol{g}_{\text{AR-B}}(\cdot, \boldsymbol{\theta}^*_{\text{AR-B}}(\alpha_1))), \end{aligned} \tag{3}$$

where models are trained by training dataset α_1 and tested by testing dataset α_2.

In this paper, we try to compare the performance of two architectures when part of dataset information is uncertain, which aims to find the probable situations that AR-A (or AR-B) performs better, corresponding to a small $C(\text{AR-A}, \text{AR-B}, \alpha_1, \alpha_2)$ (or $C(\text{AR-B}, \text{AR-A}, \alpha_1, \alpha_2)$) value.

B. Calculation and Loss Function

We start with the illustration of loss functions, which are important to search for dataset parameters in backpropagation. As shown in Fig. 1, by optimizing loss functions, we could search for the probable dataset that architecture AR-A performs better than AR-B. Then, the comparison score can be used to test the performance of a searched dataset.

In regression task, L_2 or mean square error loss is a widely used loss function to train deep learning models, where a small L_2 loss means a small error value. The L_2 loss function to train an architecture AR-A with dataset α can be denoted by:

$$L_2(\text{AR-A}, \alpha) = \frac{1}{nK}\sum_{i=1}^{n} \|\boldsymbol{g}_{\text{AR-A}}(\boldsymbol{X}_i, \boldsymbol{\theta}_{\text{AR-A}}) - \boldsymbol{Y}_i\|_2^2, \tag{4}$$

where $(\boldsymbol{X}_i, \boldsymbol{Y}_i) \in \alpha$, dataset α contains n samples, K is the dimension of $\boldsymbol{Y}$, $\|\cdot\|_2$ denotes the l-2 norm and $\boldsymbol{g}_{\text{AR-A}}$ is a neural network with architecture AR-A and parameters $\boldsymbol{\theta}_{\text{AR-A}}$.

When we deal with the situation that the dataset do not have label $\boldsymbol{Y}$ so the information of regression function is uncertain, we use a neural network with architecture AR-D to approximate the regression function $\boldsymbol{g}$ in Eq. (1). Then, when representing the dataset with the form of Eq. (1), L_2 loss becomes $L_2(\text{AR-A}, \alpha) = \frac{1}{nK}\sum_{i=1}^{n} \|\boldsymbol{g}_{\text{AR-A}}(\boldsymbol{X}_i, \boldsymbol{\theta}_{\text{AR-A}}) - (\boldsymbol{g}_{\text{AR-D}}(\boldsymbol{X}_i, \boldsymbol{\theta}_{\text{AR-D}}) + \boldsymbol{\epsilon}_i)\|_2^2$, where L_2 can also be used to train the variables that contain dataset information, including the parameters $\boldsymbol{\theta}_{\text{AR-D}}$, the distribution of $\boldsymbol{X}_i$ and $\boldsymbol{\epsilon}_i$.

To compare the performance, two architectures AR-A and AR-B first need to be trained with the same training dataset α_1 with L_2 loss:

$$\min L_{R1} = \min_{\boldsymbol{\theta}_{\text{AR-A}}} L_2(\text{AR-A}, \alpha_1) + \min_{\boldsymbol{\theta}_{\text{AR-B}}} L_2(\text{AR-B}, \alpha_1), \tag{5}$$

where the parameters of AR-A and AR-B reach $\boldsymbol{\theta}^*_{\text{AR-A}}$ and $\boldsymbol{\theta}^*_{\text{AR-B}}$ with minimum L_{R1} value.

If we want to find a probable dataset where AR-A is better than AR-B, we need to let the comparison score $C(\text{AR-A}, \text{AR-B}, \alpha_1, \alpha_2)$ small. Since small L_2 loss means small error value, searching a dataset for a small comparison score equals to search $\boldsymbol{\theta}_{\text{AR-D}}$, $\boldsymbol{X}$ and $\boldsymbol{\epsilon}$ with loss

$$\begin{aligned} \min L_{R2}(\text{AR-A}, \text{AR-B}) = \min_{\boldsymbol{\theta}_{\text{AR-D}}, \boldsymbol{X}, \boldsymbol{\epsilon}} (L_2(\text{AR-A}, \alpha_2) \\ - L_2(\text{AR-B}, \alpha_2)), \end{aligned} \tag{6}$$

where the parameters of AR-A and AR-B are trained by L_{R1} and not influenced by L_{R2}. As illustrated in the additional assumption, we let the dataset regression function architecture AR-D be the same with AR-A in L_{R2} since we want to search for the dataset that AR-A performs well and a dataset with architecture AR-A should best fit the AR-A model.

When dataset information needs to be searched, both α_1 and α_2 obey the relation described in Eq. (1), so the dataset searched with L_{R2} loss function also influence α_1 in L_{R1} loss. Although L_2 is differentiable with the respect to the parameters related to probable dataset, the L_{R1} loss only influence the parameters in AR-A and AR-B when training models. Besides, no parameters in AR-A and AR-B are influenced by the L_{R2}

loss when searching dataset. The combination of L_{R1} and L_{R2} losses means searching the dataset that a trained AR-A model performs better than a trained AR-B model in testing dataset.

$\boldsymbol{X}$ distribution, regression function $\boldsymbol{g}_{\text{AR-D}}$ and $\boldsymbol{\epsilon}$ are three unknown variables in Eq. (1) that can be searched in our method. In this paper, the dataset searching method can be applied to different types of dataset information missing. For example, when the dataset only does not have label $\boldsymbol{Y}$, the $\boldsymbol{X}_i$ in loss functions uses the known dataset input information and only parameters $\boldsymbol{\theta}_{\text{AR-D}}$ and $\boldsymbol{\epsilon}$ needs to be searched in Eq. (6). If the dataset does not have label $\boldsymbol{Y}$ and complete $\boldsymbol{X}$ distribution, $\boldsymbol{X}$ distribution also needs to be searched in Eq. (6). Concretely, the parameters $\boldsymbol{\theta}_{\text{AR-D}}$ in regression function $\boldsymbol{g}_{\text{AR-D}}$ can be trained by backpropagation algorithm since L_{R2} is differentiable with the respect to $\boldsymbol{\theta}_{\text{AR-D}}$ in α_2. If $\boldsymbol{X}$ distribution needs to be searched, we use a set of discrete input samples β_X as an approximation to represent the distribution of $\boldsymbol{X}$, where the elements of β_X obey the distribution of $\boldsymbol{X}$. The size of set β_X should be a large number (for example, 100,000) for a fine approximation. Then input samples $\boldsymbol{X}_i$ in datasets α_1, α_2 can be generated by randomly selecting inputs $\boldsymbol{X}$ from β_X and corresponding $\boldsymbol{Y}$ can be calculated with Eq. (1), where the size of α_1 is set to a fixed number to discover the relationship between training size and architecture difference. The distribution of $\boldsymbol{X}$ can be searched by adjusting the input samples $\boldsymbol{X}_i$ in set β_X through the backpropagation of input samples $\boldsymbol{X}_i$ in dataset α_2 with L_{R2} loss, where α_2 is generated from the set β_X repeatedly. Similarly, the distribution of $\boldsymbol{\epsilon}$ can be trained by adjusting the value in a finite set as an approximation.

When training the distribution of $\boldsymbol{X}$ in β_X, we need an additional normalization procedure since current calculation environment can not deal with number with large out of range value (for example, single or double precision floating point format both have maximum value). As a result, when searching $\boldsymbol{X}$ distribution, we apply a normalization method, which let each dimension in $\boldsymbol{X}$ has 0 mean and 1 standard deviation. The normalization equals to a linear operation to each dimension in $\boldsymbol{X}$. For distribution of $\boldsymbol{\epsilon}$, we normalize $\boldsymbol{\epsilon}$ with 0 mean and a small standard deviation since the expectation of $\boldsymbol{\epsilon}$ should be 0 and we aim to study the situation that the regression function has the main influence on the value of $\boldsymbol{Y}$.

Similarly, the searched output value of regression function should not out of range, so we normalize the searched regression function $\boldsymbol{g}_{\text{AR-D}}$ with loss

$$L_{RN} = \|\text{mean}(\text{AR-D}, \alpha_2) - \mathbf{0}\|_2^2 + \|\text{std}(\text{AR-D}, \alpha_2) - \mathbf{1}\|_2^2, \tag{7}$$

where $\text{mean}(\text{AR-D}, \alpha_2)$ (or $\text{std}(\text{AR-D}, \alpha_2)$) means calculating the mean (or standard deviation) vector of each dimension in $\boldsymbol{g}_{\text{AR-D}}$ on all input samples $\boldsymbol{X}_i$ in dataset α_2. L_{RN} aims to normalize the output of regression function with 0 mean and 1 standard deviation to avoid out of range large number in searching process.

In implementation, we search dataset through loss L_{R1}, L_{R2} and L_{RN} with several iterations, where in each iteration, the parameters in architectures AR-A and AR-B are randomly initialized and trained. The searched dataset will be tested by the score $C(\text{AR-A}, \text{AR-B}, \alpha_1, \alpha_2)$ for evaluation.

III. Experiments

Here, we want to evaluate several methods in comparing two architecture hyper parameters without knowing complete data information. These methods aim to save the time of adjusting hyper parameters after obtaining the complete data information. When comparing two hyper parameters denoted by architectures AR-A and AR-B with uncertain dataset information, different comparison predictions lead to different training preferences after complete data information is obtained. There are three types of preferences: 1, only training AR-A with complete data. 2, only training AR-B. 3, training both and selecting the better. Three actually training preferences correspond to different comparison prediction explanations: Corresponding to type 1 (or 2) preference, architecture AR-A (or AR-B) is always better than or similar to another in probable datasets, so only one architecture needs to be actually trained. Corresponding to type 3 preference, architecture AR-A is sometimes better and sometimes worse than AR-B in probable datasets. Type 1 and 2 comparisons adjust and select hyper parameters before the complete data information is obtained while type 3 comparison is the most conservative prediction that adjusts hyper parameters after obtaining the complete data information.

There are two key metrics for evaluating the performance of a comparison prediction: correctness and actual training times. Our goal is to pre-select an architecture before acquiring the complete dataset, and such a selection can be either correct or wrong. If a method predicts that architecture AR-A outperforms AR-B, and this holds true with the full dataset, it is considered correct; otherwise, it is wrong. Additionally, if there is no statistically significant difference between AR-A and AR-B on the complete dataset, then any pre-selection is deemed correct. For significance testing, we apply a 5% significance level t-test score based on 10 experimental runs.

Preferences Types 1 and 2 involve choosing one architecture over another, leading to outcomes that can be either correct or wrong. In contrast, Type 3 preference involves trying both architectures on the complete dataset, which always results in a correct decision but at the cost of doubling the number of training times compared to Types 1 and 2.

When considering two metrics, correct prediction times is larger the better and actually training times is smaller the better since we want to train models with smaller time consumption of adjusting hyper parameters after obtaining complete dataset information. We apply an Average Evaluation Score (AES) to consider two metrics together when making multiple comparison predictions:

$$\text{AES} = \frac{1}{N}(\text{Correct} - 2\text{Wrong} - 0.5\text{Training}), \tag{8}$$

where N is comparison predictions number. Correct (or Wrong) refers to the correct (or wrong) prediction times. Training refers to the total actually training times in the comparison predictions. In AES, the weight of training times is half of the correct

prediction times to let the most conservative type 3 prediction has 0 evaluation score. The weight of wrong prediction times is twice of the correct prediction times since a wrong comparison prediction means selecting a significantly worse architecture. Then, it needs more time and computing resources to discover and rectify a wrong selection. AES is larger the better.

1) Datasets and Hyper Parameter Candidates: We test the performance of different comparison prediction methods on time series forecasting datasets, where the time series value of a future time step is predicted by a multidimensional time series sequence. In time series forecasting, time step values of different time intervals can be predicted by the same sequence, for example, a sequence from time step 1 to 100 can be used to predict the value of time step 101, 102, 103. Then, using the series to predict the value of different time intervals are the datasets with the same input $\boldsymbol{X}$ and different label $\boldsymbol{Y}$.

Then, the experiments aim to predict the selection of different Long Short Term Memory (LSTM) hidden dimension numbers, where LSTM [8] is a commonly used architecture for time series forecasting datasets and the hidden dimension number is a commonly adjusted hyper parameters in LSTM network. Probably the best hidden dimension numbers are related to the input dimension, so we study the series with a fixed 6 dimension and 100 series length. The candidate hidden dimension numbers are 4, 32, 256, which correspond to the hidden dimension similar to the time series dimension, the hidden dimension a few times larger than the time series dimension, the hidden dimension much larger than the time series dimension. We use the last hidden state of LSTM to predict regression output by a 2-layer fully connected neural network with 50 hidden units and ReLU activation function. There are two comparison prediction tasks, where the first makes hyper parameter selection between hidden dimension number 32 and 256, the second makes selection between dimension number 4 and 32.

The experiments are made on nine different time series data with three data domains. Air: Three air quality data contain time series of daily air quality indexes collected from 1 Jan. 2014 to 1 Mar. 2022 in the Shanghai, Beijing and Shenzhen cities of China[1]. WormMotion: Three organism motion data correspond to three mutant types (wild, goa-1, unc-38) of time series extracted from EigenWorms dataset in UEA time series classification archive [9]. HandMEG: Three brain activity data contain the Magnetoencephalography (MEG) time series with three types of hand and wrist movement (left, right, up) extracted from HandMovementDirection dataset in UEA time series classification archive, where we select first 6 dimensions from 10 for forecasting to since we want to keep the ratio between data dimension number and candidate LSTM hidden dimension numbers consistent in different data. For each data, forecasting the time series with 1,2,3 and 4 intervals corresponds to four regression datasets with the same $\boldsymbol{X}$ and different $\boldsymbol{Y}$. More training samples usually has positive influence on model performance, but whether larger training dataset size could provide more concise architecture comparison prediction is uncertain. So we make experiments on various train size, from 200, 400, 800 to 1600. The validation and testing dataset size are 500. As a result, there are total 144 datasets in the experiments.

2) Baselines: • CP: Conservative Prediction (CP) does not make any preference between two architectures, so conservative prediction always make type 3 prediction. CP does not require any dataset information.

• SDP 1: Similar Distribution Prediction 1 (SDP 1) comes from the simple idea that the hyper parameter performs well in a dataset will also performs well in a similar dataset. Here, we apply SDP 1 to predict the hyper parameter of a dataset named dataset 2 based on another dataset 1, where two datasets have exactly the same $\boldsymbol{X}$ and different label $\boldsymbol{Y}$ (here dataset 1 and 2 are datasets with different forecasting intervals extracted by the same time series data). SDP 1 predicts hyper parameter of dataset 2 based on statistical significance on dataset 1, which makes type 1 (or type 2) prediction on dataset 2 if a hyper parameter named AR-A performs significantly better (or worse) on data 1. If two hyper parameters do not have significantly difference on dataset 1, SDP 1 will make type 3 prediction on data 2.

• SDP 2: Similar Distribution Prediction 2 (SDP 2) needs the same dataset information as SDP 1 and makes bold prediction based on error difference on data 1. SDP 2 makes type 1 (or type 2) prediction on dataset 2 if AR-A performs better (or worse) on data 1, no matter whether the performance is significant or not.

In our method, architecture comparison prediction can be made by searching probable datasets with loss in Eq. (6). When comparing the architecture hyper parameters denoted by AR-A and AR-B, two types of probable datasets need to be searched, including the datasets that AR-A performs better and the datasets that AR-B performs better. Then the comparison prediction is based on the searching result: If a searched probable dataset shows statistically significantly better performance of AR-A in testing dataset (after trained and validated in training and validation dataset) and no searched probable dataset shows statistically significantly better performance of AR-B, the method makes type 1 prediction. Similarly, if a probable dataset that AR-B is significantly better exists and no probable dataset that AR-A is significantly better exists, the method makes type 2 prediction. If the dataset that AR-A is significantly better and the dataset that AR-B is significantly better both can be searched, the method makes type 3 prediction. Here, we apply the probable dataset searching method to predict architecture comparison with different levels of uncertain dataset information.

• PDS 0: Probable Dataset Searching 0 (PDS 0) method searches dataset without any dataset information, so all parts of dataset information need to be searched in PDS 0.

• PDS 1: Probable Dataset Searching 1 (PDS 1) method searches dataset without any information of label $\boldsymbol{Y}$, so the regression function needs to be searched in the situation. Besides, PDS 1 has the input samples $\boldsymbol{X}$ information in the dataset but the training/validation/testing split of input

[1] https://aqicn.org/data-platform/

samples $\boldsymbol{X}$ is unknown and shuffled in PDS 1. In time series forecasting task, the true training/validation/testing split of input samples $\boldsymbol{X}$ is an important information since the input samples in each split are actually similar: In time series forecasting, input samples are extracted by a sliding window and adjacent input samples are usually put in the same split since training set should not contain the forecasting target information of validation and testing set. Then the situation that input samples in each training/validation/testing set are similar contains more information than the situation that samples in each training/validation/testing set are shuffled.

• PDS 2: Probable Dataset Searching 2 (PDS 2) method searches dataset regression function with the input samples $\boldsymbol{X}$ information, including training/validation/testing split. Similar to PDS 1, the method does not require any information of label $\boldsymbol{Y}$.

Here, different methods require different levels of dataset information, which can be ordered as follows: CP = PDS 0 < PDS 1 < PDS 2 < SDP 1 = SDP 2. SDP 1 and SDP 2 require most detailed dataset information including the input samples $\boldsymbol{X}$ information and the label $\boldsymbol{Y}$ information on a similar dataset. Probable dataset searching methods do not require such label $\boldsymbol{Y}$ information since that information usually can not be easily obtained in real world situation. Besides, CP and PDS 0 requires no specific information of the dataset.

3) Results: Table I shows the AES results of three data domains on two hyper parameter comparison tasks, with the summarized information listed in the bottom 4 lines, including average AES, average correct times, average wrong times and average actually training times. The AES result of each data domain is averaged by the AES of three data in the domain, and each data corresponds to four datasets with different forecasting intervals. For each dataset, there are other three datasets share the same input samples $\boldsymbol{X}$ and have different $\boldsymbol{Y}$, so SDP 1 and 2 methods could make three comparison predictions with each dataset. The best AES score is highlighted in bold.

Firstly, it can be found that the probable dataset searching method 2 (PDS 2) achieves the best AES score while it requires less dataset information than baselines SDP 1 and 2. The predictions made by PDS 2 could make high level of correct prediction times with small actually training times.

Secondly, the performance of PDS 0 is the same with the performance of CP, because when no dataset information is provided, each candidate hyper parameter may perform better due to the datasets searched by the method. As a result, in the experiments, PDS 0 always makes type 3 prediction for each dataset as the CP method does.

Thirdly, the difference of AES score among PDS 0, PDS 1 and PDS 2 shows that with more dataset information, the probable dataset searching method could provide better comparison prediction between hyper parameters.

Fourthly, SDP 1 and 2 perform better than CP, PDS 0 and PDS 1 since they consider more dataset information. However, SDP 1 and 2 are not practical methods because a similar dataset with the same input samples $\boldsymbol{X}$ information is hard to obtain. Besides, the predictions made by SDP 2 decrease the actually training times but increase the wrong prediction times comparing to SDP 1. Since a wrong prediction could lead to severe consequence in hyper parameter selection, SDP 2 performs worse than SDP 1 in AES score.

IV. Related Works

Uncertain Dataset Information. In application, different parts of dataset information have different difficulties to obtain, which leads to several types of task with uncertain dataset information. In unsupervised learning [1], [10], [11], [12], [13], data labels are assumed hard to be obtained so models are expected to work without data labels. In domain adaptation [2], [14], [15], [16], [17], [18], complete dataset information in source data domain and unlabeled dataset in target data domain are used to train models in target data domain. In few-shot learning [19], [20], [21], [22], [23], few train size dataset is used to train models with additional information of a similar dataset. In this paper, we discuss the adjustment of architecture hyper parameters with another kind of uncertain dataset information: Dataset labels are postponed be obtained so hyper parameters need to be adjusted without complete dataset information.

Regression Dataset Representation. Recent works [3], [4], [5], [6], [7] show that if a regression function $\boldsymbol{f_0}$ satisfies some requirements, it can be converged by a multi-layer fully connected neural network. Although the requirements are different in different works (such as Hölder function requirements [5] and (p,C)-smooth requirements [4]), they include a wide range of function types so these requirements are assumed to be satisfied by the regression function in real world functions. Our searching method use a neural network to approximate function $\boldsymbol{f_0}$ and assume that the convergence is also true in other types of neural network. Our regression dataset representation further assumes that at least a compared architecture has competitive performance when the compared architectures worth for application.

V. Conclusion

In this paper, we discuss adjusting hyper parameters with uncertain dataset information: the postponed dataset labels. To infer the probable complete dataset information, we propose a probable dataset searching method which searches the variables in our dataset representation. We test the performance of our method in 9 real world datasets.

References

[1] H. B. Barlow, "Unsupervised learning," Neural computation, vol. 1, no. 3, pp. 295–311, 1989.
[2] M. Wang and W. Deng, "Deep visual domain adaptation: A survey," Neurocomputing, vol. 312, pp. 135–153, 2018.
[3] M. Kohler and S. Langer, "On the rate of convergence of fully connected deep neural network regression estimates," The Annals of Statistics, vol. 49, no. 4, pp. 2231–2249, 2021.
[4] B. Bauer and M. Kohler, "On deep learning as a remedy for the curse of dimensionality in nonparametric regression," The Annals of Statistics, vol. 47, no. 4, pp. 2261–2285, 2019.
[5] J. Schmidt-Hieber, "Nonparametric regression using deep neural networks with relu activation function," The Annals of Statistics, vol. 48, no. 4, pp. 1875–1897, 2020.

TABLE I
COMPARISON PREDICTION AES OF SEVERAL METHODS.

Comparison Task	Dataset Domains	Train Size	CP	SDP 1	SDP 2	PDS 0	PDS 1	PDS 2
32 Dim vs 256 Dim	Air	200	0	0.083	0.167	0	0	**0.5**
		400	0	0.25	0.167	0	0	**0.5**
		800	0	0.25	0.333	0	0	**0.5**
		1600	0	0.333	**0.5**	0	0	**0.5**
	WormMotion	200	0	0.417	**0.5**	0	0	0.333
		400	0	**0.375**	0.333	0	0	0.333
		800	0	0.292	**0.333**	0	0	**0.333**
		1600	0	0.208	**0.417**	0	0	0.333
	HandMEG	200	**0**	-0.125	-0.333	**0**	**0**	**0**
		400	**0**	-0.5	-0.75	**0**	**0**	**0**
		800	**0**	-0.5	-0.833	**0**	**0**	**0**
		1600	0	-0.125	-0.167	0	0	**0.167**
4 Dim vs 32 Dim	Air	200	0	0.083	0.25	0	**0.5**	**0.5**
		400	0	0.083	0.333	0	**0.5**	**0.5**
		800	0	0.083	0.417	0	**0.5**	**0.5**
		1600	0	0.125	0.167	0	**0.5**	**0.5**
	WormMotion	200	0	0.333	0.333	0	**0.5**	**0.5**
		400	0	0.375	0.333	0	**0.5**	**0.5**
		800	0	**0.5**	**0.5**	0	**0.5**	**0.5**
		1600	0	**0.5**	**0.5**	0	**0.5**	**0.5**
	HandMEG	200	0	0.208	-0.167	0	0	**0.5**
		400	0	0.25	-0.25	0	0	**0.5**
		800	0	0.458	**0.5**	0	0	**0.5**
		1600	0	**0.5**	**0.5**	0	0	**0.5**
Total AES			0	0.186	0.17	0	0.167	**0.396**
Avg. Correct			1	0.968	0.89	1	1	1
Avg. Wrong			0	0.032	0.11	0	0	0
Avg. Training Times			2	1.434	1	2	1.667	1.208

[6] T. Suzuki, "Adaptivity of deep relu network for learning in besov and mixed smooth besov spaces: optimal rate and curse of dimensionality," in International Conference on Learning Representations, 2018.

[7] M. H. Farrell, T. Liang, and S. Misra, "Deep neural networks for estimation and inference," Econometrica, vol. 89, no. 1, pp. 181–213, 2021.

[8] S. Hochreiter and J. Schmidhuber, "Long short-term memory," Neural computation, vol. 9, no. 8, pp. 1735–1780, 1997.

[9] A. Bagnall, H. A. Dau, J. Lines, M. Flynn, J. Large, A. Bostrom, P. Southam, and E. Keogh, "The uea multivariate time series classification archive, 2018," 2018.

[10] A. Creswell, R. Kabra, C. Burgess, and M. Shanahan, "Unsupervised object-based transition models for 3d partially observable environments," Advances in Neural Information Processing Systems, vol. 34, 2021.

[11] A. Baevski, W.-N. Hsu, A. Conneau, and M. Auli, "Unsupervised speech recognition," Advances in Neural Information Processing Systems, vol. 34, 2021.

[12] P. Yu, S. Xie, X. Ma, Y. Zhu, Y. N. Wu, and S.-C. Zhu, "Unsupervised foreground extraction via deep region competition," Advances in Neural Information Processing Systems, vol. 34, 2021.

[13] S. Choudhury, I. Laina, C. Rupprecht, and A. Vedaldi, "Unsupervised part discovery from contrastive reconstruction," Advances in Neural Information Processing Systems, vol. 34, 2021.

[14] J. Dong, Z. Fang, A. Liu, G. Sun, and T. Liu, "Confident anchor-induced multi-source free domain adaptation," Advances in Neural Information Processing Systems, vol. 34, 2021.

[15] W. Zellinger, N. Shepeleva, M.-C. Dinu, H. Eghbal-zadeh, H. D. Nguyen, B. Nessler, S. Pereverzyev, and B. A. Moser, "The balancing principle for parameter choice in distance-regularized domain adaptation," Advances in Neural Information Processing Systems, vol. 34, pp. 20 798–20 811, 2021.

[16] H.-Y. Chen and W.-L. Chao, "Gradual domain adaptation without indexed intermediate domains," Advances in Neural Information Processing Systems, vol. 34, 2021.

[17] F. Lv, J. Liang, K. Gong, S. Li, C. H. Liu, H. Li, D. Liu, and G. Wang, "Pareto domain adaptation," arXiv preprint arXiv:2112.04137, 2021.

[18] M. Rostami, "Lifelong domain adaptation via consolidated internal distribution," Advances in Neural Information Processing Systems, vol. 34, 2021.

[19] Y. Wang, Q. Yao, J. T. Kwok, and L. M. Ni, "Generalizing from a few examples: A survey on few-shot learning," ACM computing surveys (csur), vol. 53, no. 3, pp. 1–34, 2020.

[20] J. Snell, K. Swersky, and R. Zemel, "Prototypical networks for few-shot learning," Advances in neural information processing systems, vol. 30, 2017.

[21] R. Wang, M. Pontil, and C. Ciliberto, "The role of global labels in few-shot classification and how to infer them," Advances in Neural Information Processing Systems, vol. 34, 2021.

[22] M. Sendera, J. Tabor, A. Nowak, A. Bedychaj, M. Patacchiola, T. Trzcinski, P. Spurek, and M. Zieba, "Non-gaussian gaussian processes for few-shot regression," Advances in Neural Information Processing Systems, vol. 34, 2021.

[23] Y. Cao, J. Wang, Y. Jin, T. Wu, K. Chen, Z. Liu, and D. Lin, "Few-shot object detection via association and discrimination," Advances in Neural Information Processing Systems, vol. 34, 2021.