A Platform-based Framework for the NAS Performance Prediction Challenge

Haocheng Wang[1], Yuxin Shen[1], Zifeng Yu, Guoming Sun, Xiaoxing Chen, Chenhan Tsai
Shanghai QXiS Company Limited.
{ericwang, evanshen, griffinyu, novasun, zenithchen, hanstsai} @qxi-smart.com

Abstract

Using few shot learning is an effective way to solve large computing resources problems in NAS sub-net performance prediction. In this study, we proposed a platform-based framework to deal with few shot problems. We explored useful information from between-training-set, between-testing-set and between-16-layer information, and further created new diverse features from original ones. By using these useful information, we designed a network model architecture with one simple, interpretable primary model and two different supporting models. Results show that RMSE of primary, supporting 1 and supporting 2 models is 0.20037, 0.2196 and 0.208, respectively. After ensembling these diverse models, the final result is greatly improved to be 0.17924. The platform-based framework can be further extended by integrating more different models to achieve better prediction precision.

1. Introduction

Neural Architecture search (NAS) is an effective way to obtain excellent networks according to the actual hardware conditions. It is well known that we have to consume a lot of computing resources to evaluate the performance of the sub-networks. Some scholars used agent tasks to predict the performance to save computing resources, which brings the gap between the predicted performance and the actual performance, so scholars have been troubled in this problem so far. Due to some correlations when using agent tasks and non agent tasks, we can use a little non agent tasks samples to improve the agent tasks prediction performance. Recently, GPNAS can be used in this problem but the performance is not perfect [1].

On the other hand, machine learning is often hampered when the data set is small. Few-Shot Learning (FSL) is proposed to tackle this problem but the core issue in FSL is that the empirical risk minimizer is unreliable [2,3]. However, most of the few shot learning research focused on deep learning or classification, and there are few solutions to the machine learning regression problem [4].

The 2021 NAS performance prediction challenge takes the above problems into consideration. The challenge uses the Mobilenet-like search space, where 16 blocks are searchable. Each block has six different operations: three choices of kernel size and two choices of expansion rate. There are two training sets provided: stages1 has 200 inaccurate samples and stage2 has 31 accurate ones. Contestants need to predict performance of test set samples according to the two training sets, that is, to solve the few shot learning problem.

2. Problem Analysis

To solve this program, we analyzed it in detail firstly. The main challenge is few-shot. There are only 231 training set data in total, and even only 31 of them are accurate. It is difficult to get good information from the provided training sets directly. Besides, even if we utilize 231 data to train some models, the overfitting issue is still very serious. To dig out the potential information from provided training sets, and to improve the overfitting issue as much as possible, we proposed a platform-based framework. It has two main parts: Information Exploration is to find useful information from given data, and Network Model Architecture is designed to reduce the overfitting issue. Both of them have expandability to accumulate the benefit from trials. With the platform and proper arrangement, the more experiments we take, the better results we get. Figure 1 shows the proposed platform-based framework and the details will be mentioned below.

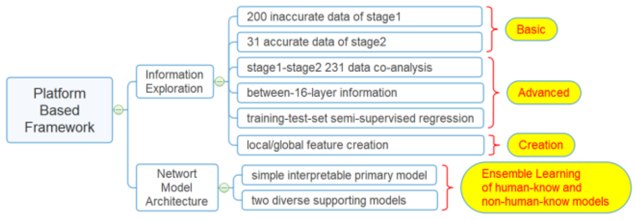
Platform Based Framework
Information Exploration
200 inaccurate data of stage1
31 accurate data of stage2
Basic
stage1-stage2 231 data co-analysis
between-16-layer information
training-test-set semi-supervised regression
Advanced
local/global feature creation
Creation
Networt Model Architecture
simple interpretable primary model
two diverse supporting models
Ensemble Learning of human-know and non-human-know models

Figure 1: Proposed Platform-based Framework

2.1. Platform-based Framework

2.1.1 Information Exploration

Since there are few training samples, each of them is precious. We try to explore useful information from the given samples. We get information from three parts: Basic part insists of the two provided training data sets. In the advanced part we explore useful information from between-training-set, between-testing-set and between-16-layer. Besides, we further re-generate new features from the original 16-layer-6-choise features. By this Feature-Creation part, the new features are much different from original ones. This diversity brings additional benefit in reducing overfitting, if we merge several models trained from diverse features.

Advanced part

Between-Training-Set: Stage1+Stage2

The number of samples given in the second stage is even smaller, only 31. It is hoped that it can be mixed with the 200 samples in the first stage to get a larger (231) sample. We designed a mapping function to map the 200 samples in the second stage to the first stage, and the final 231 samples were obtained after merging. Later experiments are also based on this expanded sample.

Between-Training-Set: Semi supervised regression

We use Coreg semi supervised learning and try to use the unlabeled test set [7]. The method of Coreg is to evaluate the confidence of pseudo labels, so as to provide pseudo labels for unlabeled data.

We have designed two Gradient Boosted Regression Trees (GBRT) with different numbers of decision trees to cross predict the unlabeled test sets to get the pseudo labels. When using a regressor, we can predict pseudo labels in unlabeled samples. The reliability is evaluated by the following formula:

$$\delta_{x_u} = \sum_{x_i \in \Omega_u} \left(\left(y_i - h(x_i)\right)^2 - \left(y_i - h(x_i)\right)^2 \right)$$

where Ω_u is the neighbor sample set in the labeled data set L, h is GBRT, and h' is the new regressor after adding xu and pseudo label $y_u(h(x_u))$ to the training set. δxu evaluates the impact of the nearest neighbor prediction after adding false labels in the training set. It means that adding the pseudo label will increase the accuracy rate and the pseudo label is more likely to be correct with the large δ_{x_u}.

To some extent, the model will learn more different data features from the pseudo-labels in the test set that can facilitate feature fusion later.

Between-Set: Cross-Validation

Divide the small samples into two groups randomly and treat them as a new training set (larger)/validation set (smaller). An experimental analysis of these new two groups can get important information. Most of our parameters (including the weight of each layer, the proportion of each layer's influence, accDiv, …), are derived from this method.

Between-16-Layer: Layer Weightings

Beside the between-set info, we have an idea that the proportion of weight influence in each layer is not necessarily the same. In this, there should be a function that can be well expressed. Figure 2 shows various curves those we experimented to represent it, and after calculating RMSE with random grouping training/validation sets, the best representative curve is -(n ^ 1.5 / 66) + 1, where n represents the nth layer. In this sense, the prior layers are more important than the later layers.

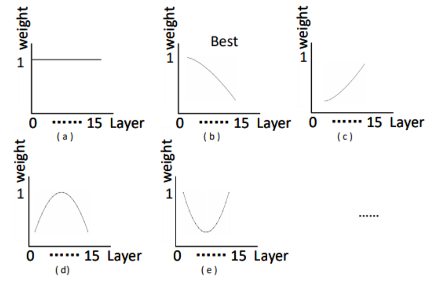

Figure 2: Importance Between Layers

Feature Creation

In NAS network search space, change of feature map and interaction relationship (in the case of sufficient training) were influenced by the order relationship between network components. We have used local/global information induction and position coding to create new features, which can express the local/global information and position information in the search space. The specific feature creation plan is as below:

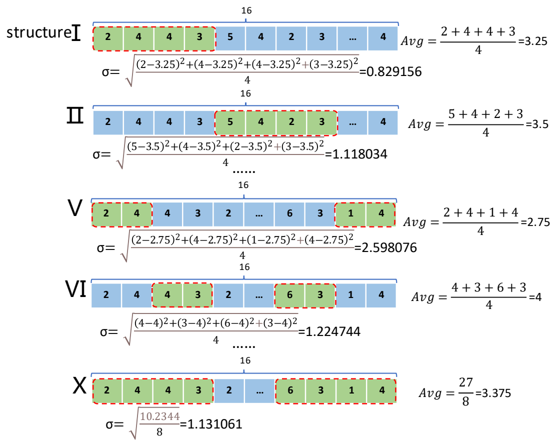

Figure 3: The process of constructing local information induction through the mean and standard deviation of feature samples at different positions

Global/Local Information Features

We can take the mean and standard deviation of n data at different positions to obtain local information (n=[4,8,16], but You can try to exhaust this value).

Position Coding and Frequency Characteristics

The sample features have position space information in the search space. We have designed a simple position encoding to create position features:

$$L = \frac{1}{n}\sum_{i=1}^{n} \cos\left(\frac{position_i(x)}{scale}\right) \quad (2\text{-}1)$$

where L is position code of the feature sample x, locate is the position of the feature sample x in the search space, scale is the scaling value, whose main purpose is to limit the distribution of position information in the cosine function.

$position$ is an integer, so the position encoding in the function distribution of cosine will not be completely equivalent, and each case has its unique distribution. Of course you can design a more complex form to express it.

We can know the distribution of sample features from the position coding in the search space [2,3]. However, the number information of samples has been discarded in coding by the averaging operation. Thus, we have created the frequency feature and added nonlinear factors to make the feature more robust.

$$R = \sigma(F(x)) \quad (2\text{-}2)$$

where R is frequency characteristic processing by non-linear method, σ is sigmoid function, and F is the frequency of characteristic value x.

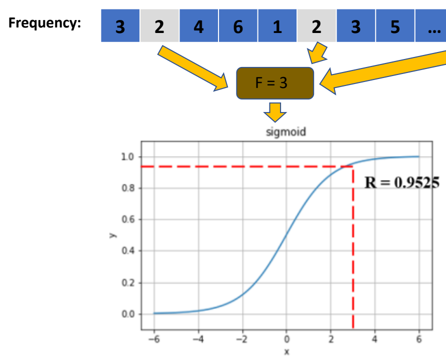

Figure 4: Samples of frequency feature construction process

Discard the original features

Convolutional Neural Network (CNN) [2, 3] has the function to extract local features of an image. In the image classification task, the image features are extracted by CNN, and then the new sparse features map can be input into the full connection layer for classification. We were inspired by CNN and manually designed the features in a interpretable way to replace the original features. The benefits of this method is that we can break out of the limitations of the original features and transform the hidden features based on the original features. The hidden features are explicitly used as model input to obtain a new model with lower correlation between the original model to make the model fusion more effectively. The experimental results show that our idea works.

2.1.2 Network Model Architecture

Primary Model

We design an interpretable model as the primary part to perform human-know features. Furthermore, we make it as simple as possible. An easy model helps reduce overfitting and let developing iteration, problem analyzing and model improvement more efficient.

According to the competition questions, a model with a weight of 16x6 was designed. Among them, 16 is the number of layers, and 6 is the 6 combinations obtained by selecting one kernel and one dilation in each layer. The three choices of the kernel should have a descending influence on the final result. The weights should be set to 1, k1, k1*k2 (where k1, k2> 1.0). Similarly, the weight of dilation can be set to 1, d1 (where d1> 1.0), and because dilation has little effect on the final result, in the experimental stage, we have further restricted the range of d1 to 1.0 <d1 <1.3. In fact, [6] consists of [1*1, k1*1, k1*k2*1, 1*d1, k1*d1, k1*k2*d1], which contains only three variables. There are 720 kinds of arrangements in [6], and the arrangement of each layer is not necessarily the same.

After the weight calculation of [16] [6], the final score is obtained. Finally, the score needs to be converted to accuracy, and a mapping function is designed to do the conversion. In the process of data analysis, I have done many experiments and found that the median is more suitable to represent the intermediate value than the average. The mapping function is shown in figure 4.

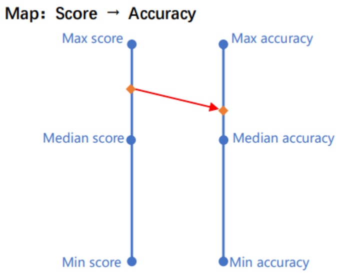

Figure 5: Score to accuracy mapping

In the early stage of participating in the competition, due to the model parameters and weight values, they have not been well adjusted. Therefore, we have added accDiv(1<=accDiv). The purpose of this parameter is to make the predicted accuracy closer to the intermediate value, and to have a smaller RMSE when the model is not very accurate. In addition, as the model becomes better and better, accDiv can be gradually reduced, and the RMSE can be further reduced.

Supporting Model

Few shot regression is the essential part of the predictor performance. Therefore, we chose Gradient Boosting regression Tree (GBRT) as the base model. GBRT passes multiple rounds of iteration, and each round of iteration produces a weak regressor, and each classifier is trained on the basis of the residual of the previous round of regressor.

Supporting 1 model: trained from data generated from Semi supervised regression (Between-testing-set).

Supporting 2 model: trained from data generated from feature creation.

Ensemble Learning

Combining multiple different models can improve the final effect. We chose a primary model with strong interpretability, plus several supporting models with as different methods as possible, for model fusion. Among them, we focus on primary model (ratio>=0.5), and supporting models as a supplementary method for fusion according to the set ratio.

There were several reasons for choosing the primary model as the main model in the beginning. One is that we chose a simple primary model, which is easier than a complex model to overfit, and because of its good interpretability, the effect is not too bad. In the auxiliary model, we use different features (local/global, position and frequency) as much as possible to create an auxiliary model with a larger difference from the main model. Since the models are more different from each other, after fusion, good results can be achieved.

After experimentation, it was found that the ratio of fusion was similar to our original idea. When the primary model is selected as the main (larger ratio), there will always be better results.

3. Experiment Results

In primary model 1 and 2, the accDiv parameters are different (1.4, 1). The positions of the 6 weights in each layer are fixed, and the weight values of each layer are slightly different. In primary model 3 and 4, the accDiv parameters are different (1.4, 1), and the positions of the weights of each layer can be different, and RMSE is the best case. The four primary models are all based on the same model concept, but the detailed parameter settings are different. After ensemble primary models with each other, they can still be regarded as the same model. There are two supporting models, which are based on the semi-supervised and GBRT models mentioned earlier.

During the experiment of the competition, we have submitted each single model to know the accuracy. From the accuracy information of the single model, we can determine the fusion ratio better. After fusing the selected models, tune parameters, the corrected accuracy can be obtained.

The following table lists some important model accuracy and the proportion of each primary model and supporting model.

Table1: Experimental Results

Primary Model		Supporting Model		Accuracy	Remark
QTY	Ratio	QTY	Ratio		
1	1	0	0	0.20037	Fix 6 weights position(each layer)
1	1	0	0	0.22629	Fix 6 weights position(each layer)
1	1	0	0	0.21948	6 weights position can be different(each layer)
1	1	0	0	0.21888	6 weights position can be different(each layer)
0	0	1	1	0.2196	Black box1(coreg)
0	0	1	1	0.208	Black box2(GBRT)
2	0.7	1	0.3	0.18784	Ensemble Model(primary+coreg)
4	0.76	1	0.24	0.185	Ensemble Model(primary+coreg)
4	0.608	2	0.392	0.17924	Best rmse in leadboard A (primary+coreg+GBRT)
4	0.504	2	0.496	0.18043	Last rmse in leadboard A (primary+coreg+GBRT)

4. Conclusion and future work

For this few shot problem, we deeply explored information from given limited data sets. By using this, we design a network model architecture, which has one primary model and two supporting models. The primary model is designed to be interpretable to perform human-know features of this contest problem, and the supporting models utilizes diverse information explored to perform non-human-know features. Our platform-based framework ensembles these mutually complement models to cover as many aspects as possible. The experiment results shows that the RMSE of primary, supporting 1, supporting 2 and final ensembled model is about 0.20, 0.22, 0.21 and 0.18, respectively.

By our platform-based framework, we can further explore more information by creating more features, and use them to train more supporting models to improve final performance.

References

[1] Li Z, Xi T, and Deng J. Gp-nas: Gaussian process based neural architecture search. Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 11933-11942, 2020.

[2] Miller E G, Matsakis N E, and Viola P A. Learning from one example through shared densities on transforms. In CVPR, 2000. 1.

[3] Fei-Fei L, Fergus R, Perona P. One-shot learning of object categories. IEEE transactions on pattern analysis and machine intelligence, 28(4): 594-611,2006.

[4] Wang Y, Yao Q, and Kwok J T. Generalizing from a few examples: A survey on few-shot learning. ACM Computing Surveys (CSUR), 53(3): 1-34, 2020.

[5] Krizhevsky A, Sutskever I, and Hinton G E. Imagenet classification with deep convolutional neural networks. Advances in neural information processing systems, 25: 1097-1105, 2012.

[6] Simonyan K, and Zisserman A. Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556, 2014.

[7] Zhou Z H, and Li M. Semisupervised regression with cotraining-style algorithms. IEEE Transactions on Knowledge and Data Engineering, 19(11): 1479-1493, 2007