A Survey of Performance Prediction

Neural Network Performance Prediction significantly reduce the tremendous cost required by training process, thus enable automatic hyper-parameter tuning. While Human Experts are adept at recognizing suboptimal model, however, there are limited works about performance prediction. *Neural Network Performance Prediction* is a advanced and promising research topic.

1 Features for Performance Prediction

A human expect may predict performance from several components: 1). Its architecture 2). Its partially observed learning curve. 3). Statistic of weights and biases along the time. Both maybe hand-crafted features or learned features can be used to predict performance of Neural Network.

1.1 Statistic from Parameters

While Statistic from *feature maps* are data-dependent and qualitative, statistic from *parameters*(including weights and biases) are more stable and general. Sinha et. al. [4] conclude that parameter evolution pattern can be generalized from one dataset to another and from one layer to another.

Sinha et. al. [4] conclude some intuition that human experts may obtain in Section 3.0:

- A very small subset of the all weights undergo massive changes compared to the rest.
- Most of the weights do not undergo a significant oscillations in value during training.

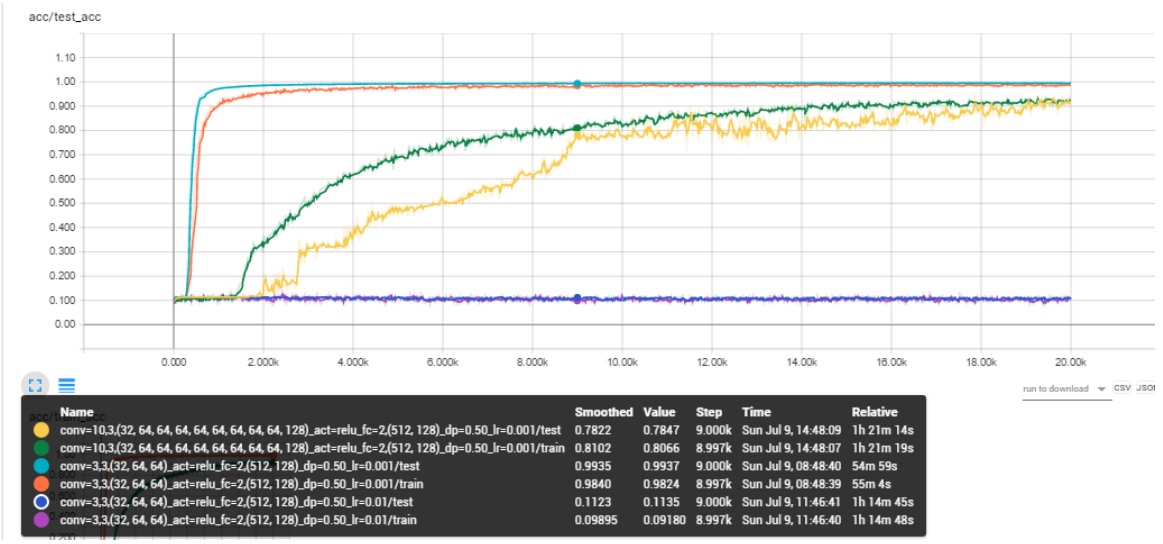
However there are no more further insights about parameter evolution pattern.

From experiments on Mnist, we observe some simple statistics like Mode/Reduced Mean/Reduced Std. Dev from histogram of parameters and conclude that:

- There are some correlations between evolution of performance and parameters. And biases also contains many informations. Ref. to fig. 1.
- With a pathological learning rate 10^{-2}, accuracy keep constant while parameters are locked in the same pattern. Ref. to fig. 1b.
- The weights of deeper layers distribute in single-guassian mode, while that of first layer distribute in Gaussian mixture mode. Thus, we may need *different* forecasting networks at different layers. However, the evolution pattern along the time in different layers are similar. Thus, we may share forecasting networks across layers. Ref. to fig. 1c.

Admittedly, the evolution patterns of parameters and correlations with performance are subtle and need to be captured by some non-linear function. [4] use a simple 1-layered Feed-forward Neural Net but leave many questions unanswered:

- How can architecture and hyper-parameter of forecasting network be determined(tuned) persuasively? ([2] use random search to find best SVM to predict performance)
- How to deal with arbitrary long time series from training history, instead of sampled at $0, 4t/10, 7t/10, t$?
- How to take relation between the weights into consideration for joint prediction, instead of predict the weights(performance for our task) independently?



(a) **Red Line:** LeNet Baseline (Train); **Purple:** Large Learning Rate; **Green:** 10 conv layers

(b) Parameters of Model with pathological learning rate almost no change

(c) Compared with fig. 1b weights are more smoothed and Gaussian-liked

Figure 1: Experiments on Mnist

1.2 Suitability for a Linear Classifier

Alain et. al. [1] re-define information as Suitability for a Linear Classifier/ Separability of Representation/ Computation Complexity rather than Entropy. Though expressed in different means, it can be viewed as prediction error using the features from intermediate layer.

The dynamic of probed signals can also be considered as hand-crafted feature for performance prediction. [1] gives some pathological example of Neural Network whose strangeness can be easily identified from dynamics of probed signals.

Meanwhile, it seems that the probed signal still cannot be explained clearly. For example, if the prediction errors are as what shown in fig. 2, then we may declare that classification using features from below branch is enough(*i.e.* information only comes from below branch and above branch is *useless*.). However, Alain et. al. argues that features come from above branch is *useful*, because the prediction error may be noisy and loss can still reduce with concatenated feature. Admittedly, combined features tend to be useful(or may be redundant), but how can it be conclude from probed signals convincingly? The weakness of probed signal lies in the locality and non-optimality nature, thus hand-crafted feature may not be the best way to observe subtle dynamics.

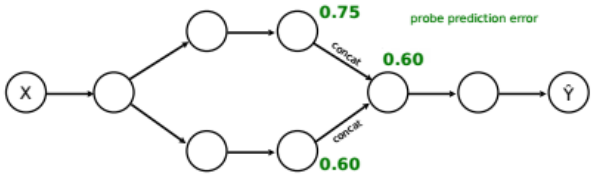


Figure 2: Predicion Error of **Above branch**: 0.75, **Below branch**: 0.60, **Concatenated**: 0.60

1.3 Architecture & Hyper-parameter

While the general representation of Neural Network is Directed Acyclic Graph/Neural Fabric, Baker et. al. [2] extract hand-crafted features $u_{\mathbf{x}}$ from Neural Net Architecture $\mathbf{x}$, which are simply *total number of weights*, *type and number of layers* and *learning rate*. These features contain very limited information about Neural Net Architecture. Finding optimal numerical representation for Architecture and Hyper-parameter is of great necessity.

1.4 Learning Curve

Existed work using SVM [3], or BNN [2] to predict performance assume that there is some smooth but noisy function mapping from hyper-parameters(and Statistic, Hyper-parameter) to the objective.

SVM outperforms BNN, because BNN learns globally from both fully observed learning curves(used as training data) and partially observed learning curves(without knowing final performance, used when inference), lacks the ability to make full use of partially observed learning curves and inferences locally. And it may suffer from underfitting because of regularization and potentially wrong prior on weights.

BNN can inference flexibly without/with partially observed learning curve, and the percentage of observed learning curve need not to be fixed in advance, because BNN also use Architecture and Hyper-parameter to inference and there are some similarity between different setting. However, SVM assumes fix-length input features and need to be retrained when different percentage of observed curves given.

The weakness of using *only* learning curve lies in: view Neural Network as a black box, cannot predict precisely. Although experiments with SVM(ν-SVR) [2] shows that learning curve(validation accuracy) is the most informative on *the dataset they constructed*, statistic from weight and dynamic inside Neural Network should count a lot. The high performance in [2] may contribute to the fact that they use training and testing dataset in the same domain, *e.g.* train AlexNet on ImageNet, collect training data, and only predict the performance of AlexNet with different hyper-parameter on ImageNet.

1.5 Statistical Interactions

Main effect happens when one independent variable affects the dependent variable of interest; *Interactions* happens when multiple independent variables influence the dependent variable of interest in non-additive fashion.

Tsang et. al. [6] propose a method to detect interaction between activations with little modification to Network. First, *adjust the network architecture* to directly model main effects as separate components in the network, as illustrated in fig. 3. The righthand primary Neural Network is the original network, and interaction can be easily detect from the weights in network.

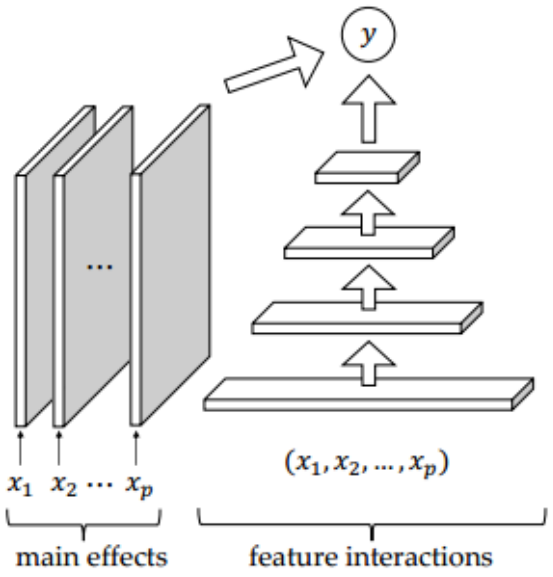


Figure 3: **Lefthand**: capture Main Effect; **Righthand**: Original Network and will be detected interaction relation

Assume i is the common neural unit, $\mathcal{I}$ is the set of interactive neural units, then the importance of interaction is evaluated as $\omega_i(\mathcal{I}) = z_i\omega(\mathcal{I}, \mathbf{W}_{i,:})$, where $\omega(\mathcal{I}, \mathbf{W}_{i,:}) = min_{j\in\mathcal{I}}\{|\mathbf{W}_{i,j}|\}$ measures the interaction strength at i and $z^{(l)} = |\mathbf{w}|^T \left|\mathbf{W}^{(L)}\right| \left|\mathbf{W}^{(L-1)}\right| \cdots \left|\mathbf{W}^{(l+1)}\right|$ measures the strength after i. Apparently, the statistical interactions only depend on weights.

Applying this method, Tsang et. al. cannot detect interaction between feature map in deep hidden layers, and they guess that neural network mainly uses its early hidden layers to model interactions and its later layers to model the nonlinearities of these interactions.

2 Necessity of Holistic Hyper-Parameter Tuning

We defines hyper-parameter as parameters that cannot be directly learned from the regular training process. They express "higher-level" properties of the model such as its complexity(Architecture of Model) or how fast it should learn(Learning Rate). While search for optimal model often concerned in Meta-modeling, we focus on *Hyper-parameter of optimization algorithm* [1] where learning rate is the most important concluded by enough tunning experiences. (Meanwhile, gradient descent optimization algorithms such as RMSprop and Adam compute adaptive learning rates for *each* parameter).

It is observed [5] that adaptive learning rate optimizer such as RMSProp and Adam tend to adjust learning rate for short-term benefits. Large learning rate may have short term negative effect and yet achieve a long term benefits. An intuitive understanding is that increasing the learning rate allows more rapid traversal of saddle point plateaus. Smith et. al. [5] use the simplest method but conduct ablation experiments. They use Cyclical Learning Rates where maximum and minimum bound and stepsize is determined from initial epochs of network training, called "LR range test".

Thus, we may need some holistic method to adjust learning rate for both convergence speed and optimal value where performance prediction shows its power.

3 Potential Future Work

- Existed work tend to use shallow model as Meta-model to predict performance. We may need to collect diverse training histories and RNN notorious for its sensitivity for Hyper-parameter is worth a try.
- Dynamic Parameters in Neural Network is of attribute size and high dimension(Spatial Dim along layer, Temporal Dim, Width, Height, Channels). Many-to-many combined with many-to-one RNN may be needed. To handle the size, we may need to note that 1). Weights of FC and Conv do not have locality pattern like images. 2). Weights in FC, RNN and Conv are quite different. 3). $S = 1, P = 0, \frac{W+F+2P}{S} + 1 = W - F + 1$ or Flatten are potential way to handle size, but not the best.

References

[1] Guillaume Alain and Yoshua Bengio. Understanding intermediate layers using linear classifier probes. *arXiv preprint arXiv:1610.01644*, 2016.

[2] Bowen Baker, Otkrist Gupta, Ramesh Raskar, and Nikhil Naik. Practical neural network performance prediction for early stopping. *arXiv preprint arXiv:1705.10823*, 2017.

[3] Aaron Klein, Stefan Falkner, Jost Tobias Springenberg, and Frank Hutter. Learning curve prediction with bayesian neural networks. 2016.

[4] Abhishek Sinha, Mausoom Sarkar, Aahitagni Mukherjee, and Balaji Krishnamurthy. Introspection: Accelerating neural network training by learning weight evolution. *arXiv preprint arXiv:1704.04959*, 2017.

[1] The hyper-parameter for optimization algorithm include Learning Rate, Momentum, Learning Rate Decay, Weight Decay/Weight Regularizer, Parameters in Weight Initialization and Initial parameter for Learning Rate and Momentum

[5] Leslie N Smith. Cyclical learning rates for training neural networks. In *Applications of Computer Vision (WACV), 2017 IEEE Winter Conference on*, pages 464–472. IEEE, 2017.

[6] Michael Tsang, Dehua Cheng, and Yan Liu. Detecting statistical interactions from neural network weights. *arXiv preprint arXiv:1705.04977*, 2017.