Exploiting Parameters Learning for Hyper-parameters Optimization in Deep Neural Networks

Michele Fraccaroli Evelina Lamma
DE - Department of Engineering
University of Ferrara, Italy
michele.fraccaroli@unife.it evelina.lamma@unife.it

Fabrizio Riguzzi
DMI - Department of Mathematics and Computer Science
University of Ferrara, Italy
fabrizio.riguzzi@unife.it

In the last years, the Hyper-parameter Optimization (HPO) research field has gained more and more attention. Many works have focused on finding the best combination of the Deep Neural Network's (DNN's) hyper-parameters (HPs) or architecture. The state-of-the-art algorithm in terms of HPO is Bayesian Optimization (BO). This is because it keeps track of past results obtained during the optimization and uses this experience to build a probabilistic model mapping HPs to a probability density of the objective function. BO builds a surrogate probabilistic model of the objective function, finds the HPs values that perform best on the surrogate model and updates it with new results. In this work, a system was developed, called Symbolic DNN-Tuner which logically evaluates the results obtained from the training and the validation phase and, by applying symbolic tuning rules, fixes the network architecture, and its HPs, therefore improving performance. Symbolic DNN-Tuner improve BO applied to DNN by adding an analysis of the results of the network on training and validation sets. This analysis is performed by exploiting rule-based programming, and in particular by using Probabilistic Logic Programming (PLP).

1 Introduction

This work aims to create a system to drive the training of DNNs by automatizing the choice of HPs analyzing the performance of each training done during the optimization process. This new system (Symbolic DNN-Tuner) extends BO [?, ?, ?] exploiting PLP [?] by mapping some tricks usually used in manual HPs optimization [?] into non-deterministic, and probabilistic Symbolic Tuning Rules (STRs). These rules specify Tuning Actions (TAs) which have the purpose of editing the HPs search space, adding new HPs or updating the network structure without human intervention. Each STR has a weight that determines the probability of application of its TA. At the beginning of the optimization process, the probabilistic weights of STRs are set manually, and then they are refined, after each training, via Learning from Interpretations (LFI) [?] (an inference algorithm available in PLP) based on the improvements obtained or not, for each TA applied in previous training.

The two main parts of Symbolic DNN-Tuner [?, ?] are a first one called Neural Block which manages all aspects regarding the networks (training, validation, HPs search space and application of the TA) and a second one called Symbolic Block that, based on the network performance and computed metrics after each training, diagnoses problems and identifies the (most probable) TA to be applied on the network architecture.

Submitted to:
SOS 2007

2 Symbolic DNN-Tuner

As mentioned before, Symbolic DNN-Tuner is a system to drive the training of a DNN, analysing the performance of each training through PLP and automatizing the choice of HPs to obtain a network with better performance. By the analysis of the metrics and the performance of the network, it is possible to identify many problems (e.g., overfitting, underfitting, etc.) that BO is not able to avoid because it works only with a single metric (validation loss or accuracy, training loss or accuracy). When Symbolic DNN-Tuner diagnoses these problems, it changes the search space of HP values or the structure of the network by applying TAs to drive the DNN to a better solution.

2.1 Symbolic Block

The Symbolic Block of Symbolic DNN-Tuner is concerned with the analysis of metrics and performance of the trained network. This part is fully developed with ProbLog [**?**]. The two major contributions of this part are the symbolic analysis and LFI. This block analyses the network metrics like loss and accuracy during both training and validation and produces a diagnosis. From this diagnosis, exploiting the STRs, the block returns the TAs to be applied.

The Symbolic Block's ProbLog program is composed of three parts: *Facts, Diagnosis* and *Tuning*. The whole logic program is dynamically created by the union of these three parts at each iteration of the algorithm. Facts memorize metrics obtained from the Neural Block, the Diagnosis section contains the code for diagnosing the DNN's behaviour problems. The Tuning section is composed of the STRs. Through ProbLog inference, we can query this program to obtain the TAs. The extracted TAs are passed to the Neural Block and applied to the DNN structure or the HPs search space.

Listing 1: STRs in the symbolic part of Symbolic DNN-Tuner

```
0.7::action(data_augment) :- problem(overfitting).
0.3::action(decr_lr) :- problem(underfitting).
0.8::action(inc_neurons) :- problem(underfitting).
0.4::action(new_conv_layer) :- problem(underfitting).
```

Listing 2: Learning From Interpretation part of Symbolic DNN-Tuner

```
% Program
t(0.5)::action(data_augment).
t(0.2)::action(decr_lr).
t(0.85)::action(inc_neurons).
t(0.3)::action(new_conv_layer).
- - - - - - - - - - - - - - - - - - - -
% Evidence
evidence(action(data_augment), True).
evidence(action(decr_lr), False).
```

Listing 1 shows an extract of all STRs in Symbolic DNN-Tuner. The values in the head of the clauses are the probabilistic weights, the `action()` predicate identifies the TAs. Then, when a `problem()` is *True*, we take the TAs with the highest probability and the TAs (e.g., `decr_lr` if `problem(underfitting)` is `true` - decrementing learning rate in case of underfitting) are applied to the network or the HPs search space.

The probabilistic weights are learned from the experience gained from previous iterations. The experience becomes the set of training examples for the LFI program. This program (Listing 2) is composed of two parts: the program and the evidence obtained from the *ImprovementChecker* module. This module checks whether the last training is better in terms of metrics w.r.t the previously performed training. The LFI program is built dynamically at each iteration of the system. After each training, Symbolic DNN-Tuner checks the improvement of the network. The improvement is a boolean value and it is used to build the evidence. The aim is to reward with greater probability those TAs that have led to improvements. In this way, Symbolic DNN-Tuner can learn which TA was better and consequently favours it over the others.

Now, with the ProbLog program updated with the just calculated new probabilities for STRs, we can query the program for the probabilities of a given atom. For clarity purposes, we report an extract of ProbLog code in Listing 3.

Listing 3: Extract of Logic section of Symbolic DNN-Tuner

```
% FACTS -------------------------------------------------------------
a([0.0529, 0.0710, 0.6266, 0.6298, ... ]).
va([0.0191, 0.0593, 0.1797, 0.2304, ... ]).
l([4.7763, 4.2189, 3.9604, 1.8257, ... ]).
vl([5.6702, 4.4710, 2.0003, 1.9812, ...]).
itacc(0.1062).
itloss(0.4125).

% DIAGNOSIS ---------------------------------------------------------
abs2(X,Y) :- Y is abs(X).
isclose(X,Y,W) :- D is X - Y, abs2(D,D1), D1 =< W.
gap_tr_te_acc :- a(A), va(VA), last(A,LTA), last(VA,ScoreA),
                Res is LTA - ScoreA, abs2(Res,Res1), Res1 > 0.2.
...

% PROBLEMS ----------------------------------------------------------
problem(overfitting) :- gap_tr_te_acc; gap_tr_te_loss.
problem(underfitting) :- high_loss; low_acc.

% TUNING ------------------------------------------------------------
action(reg_l2) :- problem(overfitting).
0.545454545454545::action(inc_dropout) :- problem(overfitting).
0.0::action(data_augment) :- problem(overfitting).
0.3::action(decr_lr) :- problem(underfitting).
...

% QUERY -------------------------------------------------------------
query(problem(_)).
query(action(_)).
```

In Listing 3 we can see the facts, the logic code for the metrics analysis and problem identification and the STRs under `FACTS`, `DIAGNOSIS-PROBLEMS` and `TUNING` section respectively. At the end of the program we can see the queries.

The `FACTS` contains the history of the accuracy and the loss during training phase and validation phase (`a([])`, `l([])`, `va([])` and `vl([])` respectively). `itacc()` and `itloss()` are two thresholds used to diagnose underfitting.

This system was tested on three dataset: CIFAR10, CIFAR100 and CIMA dataset. The first two datasets are for benchmarking purpose and the last is a dataset provided by an Italian company called CIMA S.P.A. used as industrial, real case dataset. This dataset is composed by 3200 training images and 640 of validation images of size 256x128 divided in 16 classes. These images are facsimiles of Euro-like banknotes. The performed experiments showed that Symbolic DNN-Tuner performs better than standard Bayesian Optimization on these three datasets in terms of accuracy and loss, and also provides an explanation of the possible reasons for network malfunctioning.