Multi-task Learning Curve Forecasting Across Hyperparameter Configurations and Datasets

Shayan Jawed ✉[1] , Hadi Jomaa[1], Lars Schmidt-Thieme[1] and Josif Grabocka[2]

[1] University of Hildesheim, Hildesheim, Germany
{shayan,jomaah,schmidt-thieme}@ismll.uni-hildesheim.de
[2] University of Freiburg, Freiburg, Germany
grabocka@informatik.uni-freiburg.de

Abstract. The computational challenges arising from increasingly large search spaces in hyperparameter optimization necessitate the use of performance prediction methods. Previous works have shown that approximated performances at various levels of fidelities can efficiently early terminate sub-optimal model configurations. In this paper, we design a Sequence-to-sequence learning curve forecasting method paired with a novel objective formulation that takes into account earliness, multi-horizon and multi-target aspects. This formulation explicitly optimizes for forecasting shorter learning curves to distant horizons and regularizes the predictions with auxiliary forecasting of multiple targets like gradient statistics that are additionally collected over time. Furthermore, via embedding meta-knowledge, the model exploits latent correlations among source dataset representations and configuration trajectories which generalizes to accurately forecasting partially observed learning curves from unseen target datasets and configurations. We experimentally validate the superiority of the method to learning curve forecasting baselines and several ablations to the objective function formulation. Additional experiments showcase accelerated hyperparameter optimization culminating in near-optimal model performance.

Keywords: Neural forecasting, Learning curves, Hyperparameter optimization, Sequence-to-sequence neural networks, Multi-task learning

1 Introduction

Hyperparameter optimization is a vital process in machine learning workflows. Practitioners commonly either rely on brute-force search over long grids, or via treating the loss surface in a black-box optimization framework [15]. Even so, given the configuration evaluation times, both of these methods fail to scale for large search spaces [14]. This motivates the research problem of designing novel methods to tackle this characteristic complexity and speeding up optimization. The prominent theme has been to exploit cheap-to-evaluate fidelities (or proxies) to the actual validation metrics. The simplest example of it is of a Learning curve in iterative learning algorithms which can be considered an iterative fidelity to the final performance of the hyperparameter configuration.

Learning curve forecasting methods speed up optimization by extrapolating the performance metric from short runs to arrive at keep-or-kill decisions faster. Specific works [2, 6, 10, 13] that model this extrapolation as a fidelity have exploited the partially observed time-series from validation metrics and hyperparameter configuration features (batch size, learning rate etc.) to predict the asymptote (final performance) or forecast multiple steps ahead till the asymptote. We provide an overview of the related work in the accompanying Appendix 1. [3]

Considering neural network training, there can be several statistical properties for e.g. μ, σ for $layer_i$ associated with the weights that dynamically change throughout training and can also be modeled as fidelities to the final performance [16]. In this paper, we propose a Sequence-to-sequence learning model that can model this inherent multivariate aspect present in a multi-task learning problem setting. We propose to forecast these additional channels together with the target validation accuracy for all timesteps till the asymptote. This leads to an interesting multi-task problem formulation with a rich output space to be modeled. Specifically, we formulate the main tasks to be the future points needed to be forecasted for a target channel and in contrast, all other channel's future value predictions as auxiliary tasks. An additional aspect to the learning curve forecasting problem relates to earliness in the prediction of the learning curve. The intuition behind catering for earliness is that ideally we wish to extrapolate performance of the underlying architecture from noting only a few timesteps of its performance. On the other hand, predicting the asymptote value with input of a longer length curve is comparatively trivial and not useful, since training might have converged already. We model for this aspect in the training of the forecasting network with task-specific weighting that incentivizes early forecasting of the learning curve.

A related stream of works considers meta-learning for the purpose of sample-efficiency when proceeding with hyperparameter optimization for new datasets [7, 8, 12, 17, 18]. The intuition is to exploit past optimization runs to expedite search for new datasets. Generally, the optimization runs are first gathered in a meta-dataset; a dataset describing datasets. Besides containing multivariate time series information from the iterative optimization of various architectures on various datasets, the meta-dataset also contains descriptive statistics about the underlying individual datasets, termed as meta-features. Common meta-features include, the number of attributes, classes and the instances. Hence, a meta-dataset can provide sufficient data enabling learning of a deep neural network model like we propose above, and allow transfer learning possibilities considering forecasting of a new dataset's partial learning curve. What is normally referred to as the cold-start problem in the literature [9], can be hence tackled for a new dataset by exploiting meta-features that capture dataset relations and the underlying patterns relating different hyperparameter configuration performances across datasets [7, 12]. In summary, our core contributions can be listed as follows:

– We propose a novel Sequence-to-sequence learning curve forecasting method that incorporates various additional dynamic gradient statistics in a multi-task problem formulation.

[3] Appendix available via arXiv.

- We design and optimize a novel corresponding multi-task loss function inspired by the problem setting that enforces early forecasting and incorporates biased weighted regularization for target performance metric tasks in contrast to counterpart weaker fidelity auxiliary tasks from various gradient metrics.
- We demonstrate that the method can be meta-learned and exploit configuration and meta features that generalize forecasting across hyperparameter configurations and datasets.
- We also show that our method is capable of accelerating Hyperparameter optimization when carrying out early stopping of sub-optimal configurations when integrated with model-free and meta-learned baselines.
- A thorough ablation study grounded on rigorously validating the effect of separate building components of the proposed method. Ultimately, proving the method on whole is well-founded.

2 Problem Setting

We consider a set of datasets $\mathcal{D} \in \mathbb{R}^{P \times S \times F}$, where each of the P datasets is an independent and identical set of S samples and F features upon which a supervised classification task is defined. The datasets can be differentiated based on underlying data generating processes and different data modalities (tabular data, images etc.), however, parallels can be drawn based on a set of meta-features denoted as $\phi \in \mathbb{R}^{P \times M}$. Further, we consider a set of hyperparameter configurations $\Lambda \in \mathbb{R}^{L \times K}$, where each $\Lambda_{1:L} \in \Lambda$ is a hyperparameter configuration of K hyperparameters [4] Formally, we can define the meta-dataset $X \in \mathbb{R}^{N \times C \times T}$ as a Cartesian product $\Lambda \times \mathcal{D}$, that is the result of training Neural networks [5] with L hyperparameter configurations $\Lambda_{1:L}$ on each of the P datasets. We can describe X as the set of multivariate time-series of C metrics/channels (training loss, gradient norms, validation loss, etc.) across T epochs with $N = P \times L$. For notation ease, we assume the last channel C represents a particular metric of interest (target metric), which typically is the validation accuracy. To fix ideas, the problem definition with respect to the main-task:

Given the observed metrics from the conditioning range $[1 : \tau]$ of the n-th experiment, denoted as $X_{n,:,1:\tau} \in \mathbb{R}^{C \times \tau}$ using the slicing notation;
Given the hyperparameter configuration $\Lambda_l \in \mathbb{R}^K$ and the dataset meta-features $\phi_p \in \mathbb{R}^M$ of the n-th experiment;
Predict the value of the C-th metric (validation accuracy) at the final epoch of the n-th experiment, i.e. estimate $X_{n,C,T}$.

3 Multi-LCNet: Multivariate Multi-step Forecasting with Meta-features

Our proposed model is dubbed Multi-LCNet. It is based on the encoder-decoder framework with several auxiliary tasks of predicting multiple channels for multi-

[4] Each of the $[1 : K]$ hyperparameter is sampled from a domain of valid values.
[5] In this work, we only consider Neural networks as the algorithm class.

step ahead. We let both the encoder and decoder networks be multi-layered Gated Recurrent Unit Networks (GRUs) [5]. A basic premise of our modeling objective is to exploit the configuration and meta-feature embeddings jointly with the multivariate time-series channels. However, incorporating these embeddings is not straight-forward given the fact that the rest of the data has a natural ordering with respect to time. In order to still exploit the embeddings denoted as $\xi \in \mathbb{R}^{Q \times \tau}$ jointly with the rest of the sequence modeling, we resort to repeating the embeddings on the time-axis to form additional Q channels that are concatenated with the rest of the multivariate time-series. The GRU encoder updates the hidden state recursively in the conditioning range $[1 : \tau]$. We let all channels share the same hidden-state parameters given existing correlations.

The last hidden-state from the encoding is generally referred to as the context vector [1]. Most prior approaches linearly extrapolate for one-step ahead from the context vector maximizing one-step likelihood. However, we can exploit the context vector to initialize a decoder network for multi-step forecasting. By having another decoder network, we can forecast for an arbitrarily long horizon ahead $H \in \mathbb{N}$. In addition to granting the model, the capacity to model across a wide range of fidelities, this also has a regularization effect given the pattern leading up to the asymptote can be covered in the modeling phase.

We simply initialize the decoder network's initial hidden state by copying the context from the encoder network, and feeding in the last timestep from the conditioning range i.e. τ as its first input. The decoder network has the same hidden dimensionality and number of layers as the encoder network, making this trivially possible. However, we note the discrepancy of feeding in the ground truth element at each timestep to the encoder whereas the decoder is trained in an auto-regressive manner that is consuming its own generated output at each successive timestep to compute the next hidden state and output. The output at each timestep is a $\mathbb{R}^{C+Q}$ dimensional extrapolation from the hidden state during decoding. During decoding, the model outputs $Q < M + L$ static features $[\hat{\Lambda}_l \circ \hat{\phi}_p]$ on which a reconstruction loss is defined. We noted experimentally that reconstructing static features during decoding lead to a regularization effect, improving modeling accuracy than otherwise. Additionally, we incorporate the attention mechanism [1] which allows the decoder to focus on the entirety of encoder outputs instead of solely relying on the last encoder hidden state. We refer to Appendix 2 for a more detailed description of modeling above.

3.1 Optimizing Multi-LCNet

We have formulated the problem with respect to the main-task and explained how we can generate multivariate multi-step forecasts by modeling auxiliary tasks as well. Below, we formulate objective functions with respect to both. For simplicity, let Multi-LCNet be $f(X_{n,:,1:\tau}, \Lambda_l, \phi_p, H; \theta)$ where arguments denote availability and respective ranges and θ all learnable parameters.

Standard Objective The standard approach is to train the model that predicts the target (C-th) metric at the final epoch T after observing τ observations of

the metrics. The corresponding objective:

$$\arg\min_{\theta} \sum_{n=1}^{N} \|X_{n,C,T} - f(X_{n,:,1:\tau}, \Lambda_l, \phi_p, T; \theta)_C\|_\rho \tag{1}$$

However, we could improve this objective further, as it does not use the remaining targets $c \in \{1, ..., C-1\}$, the observations after the index τ till $T-1$ and lastly does not give more importance to the first observations. Given the practical importance associated with early decision-making regarding a hyperparameter configuration, it is important to accurately estimate the target metric after only a few epochs, otherwise convergence is already reached and curve plateaued.

An *Early, Multivariate and Multi-step* Forecasting Objective To address the aforementioned drawbacks, we can optimize Multi-LCNet's parameters using the objective listed below:

$$\arg\min_{\theta} \sum_{n=1}^{N} \sum_{c=1}^{C} \sum_{t=1}^{\tau} \sum_{z=\tau+1}^{T} w_{ctz} \parallel X_{n,c,\tau+z} - f(X_{n,:,1:\tau}, \Lambda_l, \phi_p, T; \theta)_c\|_\rho \tag{2}$$

In contrast to Eq.(1), this incorporates several additional auxiliary tasks in a weighted multi-task loss. The intuition is that these auxiliary tasks induce a strong regularization effect on the main-task learning. Differentiation between auxiliary and main-tasks is defined through task-specific weighting $w_{ctz} \in (0,1) \subset \mathbb{R}^+$ [6] [7]. These task weights are hyperparameters in the objective function formulation. Manual tuning of these weights is computationally infeasible given the number of tasks could explode in the case of predicting for a decent sized horizon in standard multivariate setting. Therefore, we propose a novel factorization of the weights customized with respect to the sub-objectives relating to inducing earliness, balancing multiple channels and their point forecasts ahead:

$$\arg\min_{\theta} \sum_{n=1}^{N} \sum_{c=1}^{C} \sum_{t=1}^{\tau} \sum_{z=\tau+1}^{T} \alpha_c \beta_t \gamma_z \parallel X_{n,c,\tau+z} - f(X_{n,:,1:\tau}, \Lambda_l, \phi_p, T; \theta)_c\|_\rho \tag{3}$$

Where $\alpha_c, \beta_t, \gamma_z \in (0,1) \subset \mathbb{R}^+$ can be chosen with regard to the following insights:

i) Predicting the target metric (validation accuracy) is more important than predicting other metrics i.e. $\alpha_C > \alpha_c$. On the other hand, $\alpha_c > 0, \forall c \in 1, ..., C$ meaning we do not want to avoid predicting the other metrics since correlated channels have a beneficial regularization effect. We emphasize that such an objective formulation is a multi-task setting, where we have a target task/metric (the validation accuracy) and a set of auxiliary tasks/metrics (training loss, gradient norms, etc.).

[6] We also normalize all meta-data in $(0,1)$ unit interval

[7] We overload the notation, in this subsection w defines task-weight

ii) Correct forecasts with few observations $\tau \ll T$ are more important than estimations close to the converged epoch $\tau \approx T$. Practically speaking, we should be able to predict the performance of a poorly-performing hyperparameter configuration Λ_l after as few epochs as possible. Therefore, the weights $\beta_{1:\tau}$ can be set as exponentially decaying, which incorporates stronger penalization towards the errors made with small t values in the objective. Concretely, $\beta_{t=1} \approx 1$ and $\beta_\tau \approx 0$.

iii) Predicting the metric values at the last epoch is more important than the next immediate epoch after τ, in the desired case when $\tau \ll T$. Therefore, the forecasts indexed higher in the prediction range $[\tau + 1 : T]$ and their corresponding loss terms need to be penalized stronger. In that regard, the horizon task weights $\gamma_{\tau+1:T}$ can be formulated with the decay rate inverted and generated similarly from the exponential function. Concretely, $\gamma_{\tau+1} \approx 0$ and $\gamma_T \approx 1$.

We make the effort to elaborate more on the earliness aspect of the objective formulation given its distinct and novel formulation. In order to model for earliness, we generate what are normally called roll-outs after each input timestep observed from the learning curves. All roll-outs are multivariate multi-step forecasts till the asymptote of the curves. During training, we set $\tau = T - 1$, to utilize the full-extent of the curves and the model subsequently generates forecasts of different H length adjusted accordingly. Once all roll-outs are made, that is when $\tau = T - 1$ we can weight the errors based on the combined weighting scheme motivated above.

Exponential Weighting The exponential weighting is defined as follows:

$$\beta_{1:\tau} = \exp\left(\frac{-|j - center|}{g}\right) \qquad (4)$$
$$g = -\left(\frac{\tau - 1}{\log(u)}\right)$$

Where, j defines the index of roll-out and *center* is the parameter defining center location of the weighting function. g defines the decay. We fix $center = 0$, u is then the fraction of window remaining at the very end, that is the weight for last indexed roll-out. Setting the value for $u \in (0, 1) \subset \mathbb{R}^+$ defines the entire set of weights $\beta_{1:\tau}$. We can generate the weights $\gamma_{\tau+1:T}$ by defining another u value, replacing τ with H and inverting the weights generated through Eq.(4).

4 Experiments[8]

4.1 Datasets, Meta-Datasets and Evaluation Protocol

Meta-Dataset We use the dataset created in [21]. Each sample contains multivariate training logs of a configuration trained on a particular underlying

[8] github.com/super-shayan/multi-lcnet; Baseline implementation details in Appendix

classification dataset. All datasets used to evaluate the configurations came from the AutoML benchmark [11], in total numbering to 35. The overall meta-level distribution can be considered diverse in terms of underlying dataset characteristics such as number of samples, features and classes. Exhaustive sets of meta-features for each dataset are also available that besides these characteristics note additional many such. We also shed light on the configuration space that is used to sample valid hyperparameter configurations through in Appendix 3. We note that all architectures are funnel-shaped feed-forward networks, defined with respect to number of layers and initial units. A total of 2000 configurations are sampled from this configuration space and trained/validated/tested on the corresponding splits of each underlying dataset for a total of 52 epochs. The resulting multivariate channels also include global and layer-wise gradient statistics (max, mean, median, norm, standard deviation, and quartiles Q10, Q25, Q75, Q90), learning rate, runtime and balanced accuracies, up-to a total 54 channels. This results into X, ϕ_p, Λ_l with shapes $(N = 70000 \times C = 54 \times T = 52)$, $(P = 35 \times M = 107)$ and $(L = 2000 \times K = 7)$ respectively. We label encoded, normalized all channels besides validation accuracy between 0 and 1 and zero-padded in case of missing values due to conditionally undefined layer-wise statistics, hyperparameters or meta-features across these tensor and matrices.

Evaluation Protocol The evaluation protocol is aligned to a realistic meta-learning setting where prior meta-data across datasets is considered available and the goal would be to warm start hyperparameter optimization for new datasets as tackled in [8, 17, 18]. In light of this, we divide the meta-dataset into meta-train, meta-validation and meta-test splits covering 25, 5 and 5 datasets each. We highlight important characteristics of the validation and test split datasets in Appendix 3. We refer the remaining 25 train split datasets and a more thorough summary of data-set characteristics to [11, 21]. We split the above noted tensor and data matrices accordingly. We now proceed to define evaluation metrics that shall quantify success from different lenses. Firstly, we rely on measuring the mean-squared-error on the prediction of the last timestep (final performance) for the target metric i.e our main-task as formulated earlier in Sec. 2. In alignment with previous works [2, 13], we judge the predictions based on $\approx 20\%$ of the curve as input. Nevertheless, as motivated earlier, for the purpose of realistic hyperparameter optimization it is necessary to quantify how early the predictions match the ground-truth asymptotic performance of the curves as well. Therefore, we also evaluate our results as the average error of all final-performance errors till observing $\approx 20\%$ of the curve as input. And for completion's sake, we report the average of all final-performance errors made till $T-1$. We also report simple Regret, the difference between optimal accuracy (precomputed from meta-test data) and one achieved over a set of trials [8, 17, 18].

4.2 Baselines

Learning Curve Baselines
Last Value [14] propagates the last observed value for H timesteps.

LCNet [13] is a Bayesian Neural network that estimates parameters and creates weighted ensembles of increasing and saturating functions from power law or sigmoidal family to model learning curves. As input, however the model only takes into account configuration features by repeating these along the time-axis and learns joint embeddings via hidden layers. Hence, straight-forward application would prevent meta-learning where we wish to forecast accuracy across datasets. In light of this, we propose an extension of this model with meta-features which we refer to as **LCNet(MF)** where we simply concatenate the configuration and meta-features before joint embeddings are learned as in the standard setting.
ν-SRM is the model from [2]. The modeling for learning curves is based on training $T-1$ many feature engineered models, where each successive ν-Support Vector Machine Regression (SVR) model takes an additional timestep of the learning curve. All $T-1$ many SVR models only predict for last timestep, the validation accuracy at T. We train and validate the baseline **SRM** and its extension with multivariate channels **SRM(M)** and with multivariate channels plus meta-features **SRM(MM)** on meta-train and meta-validation datasets.
LCRankNet [18] learns latent features for learning curves via stack of non-linear Convolutional layers and architectural embeddings via Sequence-to-sequence networks. It embeds Dataset IDs for modeling learning curves across Datasets which are however generated randomly for modeling across datasets and therefore we propose to embed meta-features instead. We focus on only the ablation reported on learning $L2$-loss based pairwise rankings, which proved to be better for early predictions across all datasets and makes learning comparable to models in this paper. We remove the Sequence-to-sequence based embeddings that might be more applicable to deeper network topologies as tackled originally in that paper. Extension of **LCRankNet** with multivariate channels is termed **LCRankNet(M)**, and like before we also craft **LCRankNet(MM)**.

Multivariate Multi-Step Forecasting Baselines
TT-RNNs Tensor-train RNN is a sequence-to-sequence model [20]. The working principle is to replace the first-order markovian dynamics abiding hidden states in RNNs with a polynomial expansion computed over the last many hidden states. Tensor decomposition is used for dimensionality reduction for this new state. We train the baseline on all meta-train datasets but unlike above baselines, repeated the static features including configuration and meta-features on the time-axis to form additional channels for **TT-RNN (MM)**.
MCNN is a multivariate time-series forecasting baseline crafted through heuristically searched parameter sharing between stacks of convolutional layers and exponentially decaying weighted schemes that tackle scale changes in long-term forecasting. We also designed another meta-feature based extension named **MCNN(MM)** where we stack non-linear embedded meta and configuration features directly with the latent convolutional features before feeding to the stack of fully-connected layers predicting for multiple channels and time-steps ahead.
Multi-Task LASSO induces shared sparsity among parameter vectors of multiple regularized linear regression models. In our setting, we can consider all

timesteps to be forecasted as separate tasks and instead of solving multiple lasso models independently, feature selection is stabilized by shared sparsity induced via block-regularization schemes.

Model-free Hyperparameter Optimization Baselines
Random Search [3] samples hyperparameter configurations randomly from the space of configurations defined in the meta-split.
Hyperband [14] is a bandit-based method that samples configurations randomly and terminates sub-optimal configurations according to predefined downsampling rates at each round, only advancing better performing ones to be run for more iterations. We report results for different downsampling rates in brackets Table. 2 with fixed max-iterations i.e. 52 from meta-data.

Meta Learning Hyperparameter Optimization Baselines For the purpose of hyperparameter optimization, most work has focused on meta learned Bayesian Optimization (BO). Hence, we benchmark and propose orthogonal extensions to:
TAF from [19], given the same intuition of ours, transfers knowledge between tasks in a meta-setting. Transferable Acquisition Function (TAF) incorporates source and target relationships in the acquisition function during BO. The acquisition function scores the next configuration based on expected improvement on the target dataset and predicted improvement over the source datasets. We also orthogonally integrate Multi-LCNet as a meta-learned forecasting model within BO. The Gaussian Process (GP) surrogate updates its parameters sequentially on early terminated estimations of Multi-LCNet instead of on final-performances of configuration trained fully. We refer to the extension as **TAF-MLCNet**. Also, early terminating only applies to meta-testing, source GPs remain unaltered.
MBO from [17] is the recent state-of-the-art baseline for Meta-learning in Bayesian Optimization (MBO) that acquires next configurations efficiently in a meta reinforcement learning setting and modeling the acquisition function with a neural network. We also orthogonally integrate Multi-LCNet similar to above crafting **MBO-MLCNet**.

Multi-LCNet Ablations
Multi-LCNet(M) does not embed meta-features. However, it uses configuration features repeated and forecasted as channels.
Multi-LCNet(MM) embeds meta-features jointly with configuration features and forms channels with these joint embeddings as noted earlier.

4.3 Forecasting Results

We report the comparison of our proposed Multi-LCNet with the learning curve and multivariate forecasting baselines in Table. 1. We ran all baselines described earlier with 3 different seeds and report mean performances across standard objective with $\tau = 9$ and the aggregated metrics *A(9),A(51)* that quantify

Table 1: Comparison against learning curve and multivariate forecasting baselines in terms of MSE $\cdot 10^{-2}$ on validation accuracy scaled to [0-1]. Columnar least is boldfaced, second-least is underlined.

Methods	Segment			Shuttle			Sylvine			Vehicle			Volkert		
	$\tau = 9$	A(9)	A(51)	$\tau = 9$	A(9)	A(51)	$\tau = 9$	A(9)	A(51)	$\tau = 9$	A(9)	A(51)	$\tau = 9$	A(9)	A(51)
LCNet(MF)	1.83	3.57	2.98	32.24	27.1	35.44	3.28	2.38	3.25	2.29	5.39	3.21	2.69	3.03	5.34
SRM	**0.95**	2.04	0.46	2.02	4.45	1.09	0.42	1.08	0.22	**0.4**	0.83	**0.19**	0.24	0.38	0.1
SRM(M)	1.37	3.21	0.69	2.55	5.35	1.38	0.87	2.58	0.5	0.67	1.24	0.29	0.2	0.39	0.1
SRM(MM)	1.29	2.75	0.6	2.9	6.07	1.43	1.05	2.51	0.52	0.63	1.22	0.28	**0.19**	**0.3**	**0.08**
LCRankNet	1	1.7	0.4	**1.24**	2.67	**0.63**	**0.27**	0.79	0.16	0.58	1.39	0.3	0.6	1.83	0.36
LCRankNet(M)	1.34	2.81	0.63	1.86	3.94	0.96	0.76	2.32	0.48	0.55	1.28	0.29	0.21	0.56	0.13
LCRankNet(MM)	0.99	1.62	0.49	1.43	2.79	0.94	0.62	1.25	0.35	0.75	1.35	0.44	0.49	0.84	0.2
Last Value	1.51	3.57	0.76	2.76	6.15	1.61	0.69	1.72	0.35	0.73	1.6	0.35	0.31	0.78	0.17
TT-RNN	1.56	2.29	–	4.95	5.72	–	0.97	1.5	–	1.15	1.79	–	1.2	2.64	–
TT-RNN(MM)	5.63	4.62	–	10.08	11.47	–	5.12	4.99	–	7.84	7.06	–	5.15	2.86	–
MCNN(M)	7.25	7.04	7.1	5.05	5.28	5.18	2.42	2.5	2.46	7.51	7.26	7.31	9.54	9.14	9.25
MCNN(MM)	7.14	7.1	7.04	5.13	5.29	5.2	2.47	2.5	2.48	7.38	7.33	7.23	9.32	9.21	9.14
MTL-LASSO(M)	1.35	1.99	–	6.32	7.85	–	1.49	1.95	–	0.96	1.31	–	0.77	0.94	–
Multi-LCNet(M)	1.02	**1.47**	**0.38**	1.57	**2.27**	1.42	**0.27**	**0.68**	**0.15**	0.92	1.53	0.38	1.46	3.27	0.7
Multi-LCNet(MM)	1.07	2.09	0.5	2.68	3.9	1.43	0.61	1.38	0.32	**0.4**	**0.75**	0.21	0.32	0.94	0.2

earliness and dynamic performance throughout the curve length. *A(9),A(51)* denote the Average of final-performance errors (main-task) made observing curves till $\tau = 9$ and $\tau = 51$ respectively. A number of interesting observations can be drawn from these results. Firstly, we can see that Multi-LCNet is able to outperform the baselines across multiple metrics on all datasets besides one. Most interesting are the lifts on *A(9)* compared to other metrics, since it quantifies performance with regard to earliness. We can credit the auxiliary supervision provided to the model through multivariate multi-step forecasting as the basis for these leads. This stands to reason, provided learning curve baselines already rely on deep learning primitives such as feed-forward layers and convolutions in LCNet and LCRankNet. Crucially, the baselines are all provided input of the same dimensionality with their respective extensions.

Another set of observations can be derived from benchmarking the performance of machine learning baselines to the naive last value forecasting baseline. We can validate the finding from prior works about this baseline's exceptionally strong performance on learning curves, which have a natural tendency to plateau rather early. Nevertheless, we can see that majority of learned baselines outperform it especially on the first two metrics that quantify earliness.

With regard to auxiliary supervision through multivariate multi-step forecasting baselines, we observe that MTL-LASSO and TT-RNN perform equally well across datasets and metrics, but indeed are less generalizable than counterpart learning curve baselines. We hypothesize that the reason for this sub-par performance is due to the rather fixed dimensionality forecasts that prohibit all multivariate forecasting baselines considered to exploit training on dynamic length input curves. These baselines were initially proposed for long range input as compared to extremely short learning curves where data cannot be generated through rolling windows and rather every curve needs to be partitioned into a fixed conditioning and prediction range beforehand. This explains why

despite MCNN and LCRankNet being both convolutional neural networks, the LCRankNet baseline and extensions can perform much better by modeling for only 1 fixed window but exploiting the same curves multiple times with dynamic conditioning history. This is also the reason why one joint model across all input length for LCNet, SRM, MTL-LASSO, TT-RNN, MCNN is not possible and we did hyperparameter tuning for validating their performance for only $\tau = 9$ and re-trained the baselines for all other $\tau = [2...51]$. We also dropped the comparison given scalability challenges with TT-RNN, MTL-LASSO for metric *A(51)* as noted by '–' in Table. 1. Except for these baselines, we tuned the hyperparameters for all other models on the metric $A(51)$. On the other hand, this highlights yet another advantage of Multi-LCNet which can be trained on dynamic conditioning history as well as exploit auxiliary regularization through multivariate forecasting as it can generate dynamic length forecasts from any window of the curve.

Additional observations can be made with regard to multivariate channels, hyperparameter configuration features and lastly meta-features. SRM baseline is unable to cater for both multivariate gradient statistics and meta-features, as evident by higher errors made throughout the metrics and datasets by respective extensions. This could be because the model is considered rather shallow and unable to learn non-linear feature interactions as the deep learning counterpart methods are able to. In fact, we see that LCRankNet and Multi-LCNet both benefit from additional multivariate information plus meta-features comparatively more so. On the other hand, we can observe that directly feeding all static features as channels through repetition on time axis lead to downgrade in modeling accuracy for both the TT-RNN and MCNN baselines. This is the reason why for Multi-LCNet we explored another way to embed meta-features to a more fine-grained representation explained earlier in the method section.

4.4 Accelerating Hyperperameter Optimization

In this section, our aim is to firstly formulate a predictive termination criterion based on the predictions from Multi-LCNet. We follow the lead of [2, 4, 6], and model a similar criteria that is essentially based on the heuristic that if at any given time the forecasted accuracy from a partially observed configuration's curve falls below a certain best observed accuracy in a given set of configurations, then this configuration can be early terminated to save valuable compute and time resources. Specifically, we adapt the criteria from [2] to a meta-setting. To ground the termination decision in probabilistic terms, one can model the forecast as a Gaussian perturbation around the original estimate to safeguard against poor out-of-sample generalization. To fix ideas, we randomly sample $M \ll N$ configurations where $m \in 1, ..., M$ and at each successive epoch τ, we generate forecasts, $\hat{X}_{1:M,C,T}$ for all M configurations. To model the uncertainty associated with the forecasts we estimate the standard deviation σ by leave-p-out cross-validation [9]. Specifically, we account for the uncertainty by keeping dropout active and noting the σ among $\{(\tau - p)...\tau\}$ forecasts of $\hat{X}_{1:M,C,T}$. With the uncertainty, we

[9] We overload notation σ to denote standard deviation, p in cross-validation

can estimate the forecasts as a gaussian perturbation $\hat{y}_{1:M,C,T} = \mathcal{N}(\hat{X}_{1:M,C,T}, \sigma)$. Finally, the probabilistic termination criterion: $p(\hat{y}_{1:M,C,T} \leq \max(X_{1:M,C,1:\tau})) = \Phi(\max(X_{1:M,C,1:\tau}); \hat{y}_{1:M,C,T}, \sigma)$, where $\Phi(.; \mu, \sigma)$ is the Cumulative distribution function (CDF) of the Normal distribution. For configuration m if probability $p(\hat{y}_{m,C,T} \leq \max(X_{1:M,C,1:\tau})) \geq \Delta$ does not hold, we can early terminate it. Where, Δ balances the tradeoff between early terminating configurations for more significant acceleration or on the other hand the risk of observing higher regret. Additionally, for ensembling's sake, one can let the top-η confs to complete training. For our experiments we set Δ, p and η via cross-validation on the validation datasets based on observing the regret. We note that this criteria despite sharing characteristic similarities differs from the one in [2], given the cross-dataset setting. This setting does not require any burn-in period to observe learning curves completely for new datasets. As a downside however, we track the maximum observed accuracy from $[1..\tau]$ for all new configurations instead of the accuracy at T from the burn-in period. Other notable differences include a more robust estimation of uncertainty given dropout and multi-horizon recursive forecasts and tuning of Δ, p and η on meta-validation. We also refer to an example termination in Appendix 4.

4.5 Acceleration Results

This section reports the results on accelerating hyperparameter optimization through early termination. For our first set of experiments we accelerate Random Search (RS-MLCNet) given its simplicity and vast utility. We randomly sample two sets of confs. i.e. M=50 and M=100 and report corresponding time(m) and regret in Table. 2. We report the average of 10 runs. We set $\delta = 0.99$, $\eta = 5$ and $p = 5$ for both Multi-LCNet(RS-MLCNet) and SRM based early stopping (RS-SRM). The regret is stated in terms of percentage classification accuracy. We first note the comparison between RS and its counterpart acceleration with Multi-LCNet. The results indicate huge gains with regard to time saved with very little to no harm in regret. We also note that the standard deviation is on a similar scale. Since initial random selection of 50 or 100 configurations from 2000 is bound to affect the final regret, we keep these same for RS, and its acceleration through Multi-LCNet & SRM across all runs.

We also compare these gains with SRM based early termination. In terms of retrieving the optimal model among the initial trials, both RS accelerations lead to similar regret given the same early stopping criteria. Our initial assumption was that the difference in MSE would result in better acceleration performance, but however in terms of regret computed for optimal configuration the gains in forecasting accuracy did not transfer gracefully.

We also compare accelerated RS (RS-MLCNet) to Hyperband. Hyperband also randomly samples configurations and uses the last value based extrapolation to early terminate. However, Hyperband dynamically selects the configurations to evaluate, which prohibits reporting results for 50 or 100 trials. To report a fair comparison, we report results for Hyperband initialized with three different

downsampling rates. Among the two trial settings for RS-MLCNet and Hyperband variants, we can see that on Segment, Shuttle and Volkert we are able to outperform Hyperband in terms of balance between regret and time taken.

Lastly, we benchmark the meta-learned approaches TAF and MBO. Firstly, we note that both these baselines stand out due to consistent minimal possible regret across both trial settings [10]. This is consistent with known superiority of Bayesian optimization to RS. The efficacy of RL framework from MBO saves even more time compared to BO based TAF. We observed that MBO can ask to run the same configuration repeatedly, in contrast to TAF if it discovers the optimal configuration early on. Hence, we count the regret and time taken for only unique configurations among the 50 or 100 specified initially [11] and can see that the difference in time for the two initial trial sets remains similar for MBO. We turn to report the results for Multi-LCNet integrated counterparts MBO-MLCNet and TAF-MLCNet that enable early termination for both these methods. Given the drawback that TAF and MBO are both sequential approaches, we modify the early stopping criteria to consider the values until current timestep for only the single incumbent configuration. This puts the early stopping criteria at a disadvantage, but nevertheless we observe a clear lift across all datasets without loss in regret. Equally worth noting is the fact that early terminated objectives do not generally interfere with acquisitions within the context of either BO nor RL. This is important because one might worry that the sequential chain of successive configuration acquisitions might be affected if the underlying GP parameters are updated on the early stopped performance (objectives) for target dataset configurations instead of on their final performance. Nevertheless, when provided with early stopped objectives, the number unique confs. did arise for MBO-MLCNet@100 for Shuttle and Sylvine datasets notably. We hypothesize this is due to comparable fewer differences between configurations on these datasets compared to other datasets as evident in lower standard deviation for RS regret too. Still, with higher number of configurations, the times were less.

4.6 Ablation Study on the Meta-Validation Set

The above objective function comprehensively captures the entirety of the multi-task output space, with sub-objectives exploiting inherent characteristics of the learning curve forecasting problem setting. However, there exist possibilities of designing the objective functions in between the two extremities, as given by Eq. (1) and Eq. (3). Specifically, if we consider either of the earliness, multi-target or the multi-step outer loop as either present or discarded leads to $2^3 = 8$ possibilities with respect to objective formulation. We can study if either of the unstated 6 combinations, for example the standard function in Eq. (1) equipped with earliness and associated hyperparameters $\beta_{1:\tau}$ dictating the exponentially

[10] The results for MBO and TAF are not averaged across runs given the stationarity of GP modeling and meta-data; based on personal correspondence with the authors.

[11] Optimization is not terminated when regret is 0 to simulate real-world testing where regret is unknown apriori.

Table 2: Accelerated Hyperparameter Optimization results.

Methods	Segment		Shuttle		Sylvine		Vehicle		Volkert	
	Time	Regret	Time	Regret	Time	Regret	Time	Regret	Time	Regret
MBO@50	35.29	0.0	164.61	0.0	40.76	0.0	25.84	0.0	75.09	0.0
MBO@100	46.43	0.0	170.5	0.0	47.86	0.0	27.38	0.0	82.18	0.0
TAF@50	78.02	0.0	186.59	0.0	66.3	0.0	72.52	0.0	268.16	0.0
TAF@100	160.77	0.0	364.68	0.0	146.86	0.0	125.33	0.0	447.14	0.0
MBO-MLCNet@50	24.39	0.0	129.29	0.0	31.9	0.0	18	0.0	52.14	0.0
MBO-MLCNet@100	30.81	0.0	145.47	0.0	36.08	0.0	18.72	0.0	68.89	0.0
TAF-MLCNet@50	43.22	0.0	117.68	0.0	55.01	0.0	47.07	0.5319	169.24	0.0
TAF-MLCNet@100	93.97	0.0	200.75	0.0	110.45	0.0	84.24	0.0	322.75	0.0
Hyperband(2)	24.9±3.2	4.3±0.7	57.±9.1	1.0±0.5	38.1±5.1	0.5±0.4	26.6±3.0	3.0±3.1	62.6±6.3	7.2±3.2
Hyperband(2.5)	15.4±2.0	4.4±1.4	32.9±3.4	1.2±0.5	20.5±1.8	0.7±0.5	15.±1.3	3.4±2.2	37.6±5.0	7.8±4.3
Hyperband(3)	9.1±0.9	5.6±0.9	22.5±5.5	1.5±0.3	16.3±3.4	0.6±0.5	10.±1.3	4.3±1.5	21.8±2.7	8.7±3.3
RS@50	57.4±4.4	5.6±2.4	126.1±11.6	0.5±0.6	60.2±7.1	1.1±0.7	53.1±5.8	6.3±1.4	188.1±24.	8.2±3.8
RS@ 100	135.5±12.5	3.3±0.7	298.6±24.1	0.4±0.2	146.4±11.7	0.4±0.4	131.±7.7	5.0±0.7	384.3±14.9	5.3±3.3
RS-SRM@50	8.0±0.7	5.6±2.4	25.7±3.9	1.1±0.7	9.±1.3	1.5±1.1	7.8±1.0	6.7±1.3	24.1±1.1	8.9±4.5
RS-SRM@100	13.2±0.8	3.4±0.9	38.6±5.3	1.0±0.6	16.2±1.5	1.1±0.9	13.2±1.1	5.0±0.7	43.3±3.5	5.3±3.3
RS-MLCNet@50	8.0±0.7	5.7±2.5	25.5±4.0	1.1±0.7	9.1±1.1	1.5±1.1	8.0±1.3	6.7±1.3	35.1±3.1	8.5±4.0
RS-MLCNet@100	13.±0.7	3.4±0.9	36.9±3.9	1.1±0.6	15.9±2.0	1.2±0.8	13.4±1.2	5.0±0.7	64.3±3.7	5.3±3.3

decaying scheme for task weights models the main-task more accurately than the objective in Eq. (3). Moreover, we can also study whether any of the associated components in the input space, channels $c \in \{1, ..., C-1\}$, the configuration features Λ_l and the meta-features ϕ_p lead to improvement or on the contrary decline in modeling accuracy with respect to the main-task. To fix ideas, we term the changes to the objective function and removal of input configuration or meta-features as ablations and refer to number of recurrent layers, number of hidden units, number of fully connected layers and respective units, activation functions, dropout, batch sizes, learning rate as standard hyperparameters to the network that need to be tuned regardless the ablation. Additionally, we consider the auxiliary task weights $\alpha_c, \beta_t, \gamma_z$ as conditional hyperparameters that are only defined when the corresponding ablation is chosen. Attention is considered as additional conditional hyperparameter that is defined only for ablations considering encoder-decoder modeling of the entire horizon. We tune the hyperparameters of each ablation together with well-defined associated hyperparameter configurations, since one hyperparameter config. might not generalize to another ablation.

For all ablations besides the proposed objective formulation and input space, we introduce another model termed as Standard-Net. This model is characterized as an encoder-only network with an output fully-connected-layer whose dimensionality corresponds to the dimensionality of the target space. The target space can vary from a single point to forecast for the main channel (as in standard objective) or all channels at the last horizon and lastly forecasting all channels for the entire horizon. Interestingly, the Standard-Net cannot be trained with an Earliness sub-objective when impeded by fixed dimensionality of output. Also, conditional upon the ablation Standard-Net can incorporate all input channels, meta-features and configuration features, but however cannot incorporate attention.

Including root-level binary valued hyperparameters that define presence or absence of ablative sub-objectives and input features, hyperparameters conditioned upon these ablations (α, β, γ) and standard network hyperparameters leads to 14-dimensional hyperparameter configurations. Given this relatively large search space, we rely on Hyperband [14], to conduct a thorough analysis in order to better judge whether the proposed formulation of Multi-LCNet leads to a

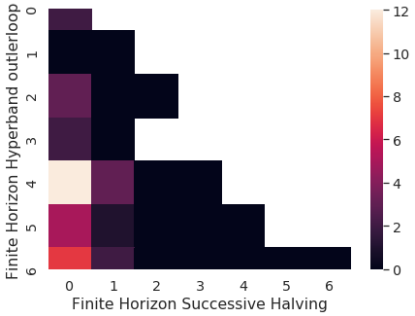

Fig. 1: In each cell we plot the ratio of StandardNet configurations selected to Multi-LCNet configurations across different successive halving iterations in Hyperband. We can observe that Hyperband increasingly selects Multi-LCNet configurations as successive halving continues and hence the ratio decreases

gain in predictive accuracy over Standard-Net. The working principle behind Hyperband also qualitatively expresses whether a particular configuration is iteratively selected consecutively in various levels of Successive Halving. We define a large search space to randomly sample configurations from and observe that configurations trained with the proposed objective formulation are given increasingly higher budget, which testifies modeling accuracy of the proposed method to be higher than counterpart ablations. We note for Hyperband that all initial search spaces and following number of successive halving rounds are defined with respect to maximum number of iterations. By setting this to 1000 and default downsampling rate (=3), we allow for the possibility of multiple rounds and larger initial search spaces before these rounds. In these spaces ablations outnumber the Multi-LCNet configurations by 8x, however, across all successive halving iterations (y-axis of Fig. 1), the ratio converges to 0 (x-axis) showcasing that Hyperband spends more budget on selected Multi-LCNet configurations

5 Conclusion

In this work, we propose a novel meta-learned forecasting model that models validation accuracy and several additional gradient statistics in a weighted multi-task loss. Empirical evaluation showed the model outperformed multiple forecasting baselines and forecasts can be used to accelerate hyperparameter optimization in the simple case of random search and also meta Bayesian optimization. As future work, we shall extend the modeling in novel meta-learning directions.

Acknowledgements

This work is co-funded by the industry project "Data-driven Mobility Services" of ISMLL and Volkswagen Financial Services; also through "IIP-Ecosphere: Next Level Ecosphere for Intelligent Industrial Production".

References

1. Bahdanau, D., Cho, K., Bengio, Y.: Neural machine translation by jointly learning to align and translate. arXiv preprint arXiv:1409.0473 (2014)

2. Baker, B., Gupta, O., Raskar, R., Naik, N.: Accelerating neural architecture search using performance prediction. arXiv preprint arXiv:1705.10823 (2017)
3. Bergstra, J., Bengio, Y.: Random search for hyper-parameter optimization. Journal of machine learning research **13**(2) (2012)
4. Chandrashekaran, A., Lane, I.R.: Speeding up hyper-parameter optimization by extrapolation of learning curves using previous builds. In: Joint European Conference on Machine Learning and Knowledge Discovery in Databases. pp. 477–492. Springer (2017)
5. Chung, J., Gulcehre, C., Cho, K., Bengio, Y.: Empirical evaluation of gated recurrent neural networks on sequence modeling. arXiv preprint arXiv:1412.3555 (2014)
6. Domhan, T., Springenberg, J.T., Hutter, F.: Speeding up automatic hyperparameter optimization of deep neural networks by extrapolation of learning curves. In: Twenty-Fourth International Joint Conference on Artificial Intelligence (2015)
7. Feurer, M., Klein, A., Eggensperger, K., Springenberg, J.T., Blum, M., Hutter, F.: Auto-sklearn: efficient and robust automated machine learning. In: Automated Machine Learning, pp. 113–134. Springer, Cham (2019)
8. Feurer, M., Letham, B., Bakshy, E.: Scalable meta-learning for bayesian optimization. arXiv preprint arXiv:1802.02219 (2018)
9. Gantner, Z., Drumond, L., Freudenthaler, C., Rendle, S., Schmidt-Thieme, L.: Learning attribute-to-feature mappings for cold-start recommendations. In: 2010 IEEE International Conference on Data Mining. pp. 176–185. IEEE (2010)
10. Gargiani, M., Klein, A., Falkner, S., Hutter, F.: Probabilistic rollouts for learning curve extrapolation across hyperparameter settings. arXiv preprint arXiv:1910.04522 (2019)
11. Gijsbers, P., LeDell, E., Thomas, J., Poirier, S., Bischl, B., Vanschoren, J.: An open source automl benchmark. arXiv preprint arXiv:1907.00909 (2019)
12. Jomaa, H.S., Schmidt-Thieme, L., Grabocka, J.: Dataset2vec: Learning dataset meta-features. arXiv preprint arXiv:1905.11063 (2019)
13. Klein, A., Falkner, S., Springenberg, J.T., Hutter, F.: Learning curve prediction with bayesian neural networks (2016)
14. Li, L., Jamieson, K., DeSalvo, G., Rostamizadeh, A., Talwalkar, A.: Hyperband: A novel bandit-based approach to hyperparameter optimization. The Journal of Machine Learning Research **18**(1), 6765–6816 (2017)
15. Shahriari, B., Swersky, K., Wang, Z., Adams, R.P., De Freitas, N.: Taking the human out of the loop: A review of bayesian optimization. Proceedings of the IEEE **104**(1), 148–175 (2015)
16. Unterthiner, T., Keysers, D., Gelly, S., Bousquet, O., Tolstikhin, I.: Predicting neural network accuracy from weights. arXiv preprint arXiv:2002.11448 (2020)
17. Volpp, M., Fröhlich, L.P., Fischer, K., Doerr, A., Falkner, S., Hutter, F., Daniel, C.: Meta-learning acquisition functions for transfer learning in bayesian optimization. arXiv preprint arXiv:1904.02642 (2019)
18. Wistuba, M., Pedapati, T.: Learning to rank learning curves. arXiv preprint arXiv:2006.03361 (2020)
19. Wistuba, M., Schilling, N., Schmidt-Thieme, L.: Scalable gaussian process-based transfer surrogates for hyperparameter optimization. Machine Learning **107**(1), 43–78 (2018)
20. Yu, R., Zheng, S., Anandkumar, A., Yue, Y.: Long-term forecasting using higher order tensor rnns. arXiv preprint arXiv:1711.00073 (2017)
21. Zimmer, L., Lindauer, M., Hutter, F.: Auto-pytorch tabular: Multi-fidelity metalearning for efficient and robust autodl. arXiv preprint arXiv:2006.13799 (2020)