Meta-learning Hyperparameter Performance Prediction with Neural Processes: Appendix

Ying Wei [1] **Peilin Zhao** [2] **Junzhou Huang** [2]

A. Additional Details for Methodology

A.1. Architectural details for TNP

We show the architectural details of TNP in Figure 1, which is adapted from ANP (Kim et al., 2019). The MLPs for the encoder, the decoder, and the embedding function g for attention are all linear in the final layer and have ReLU as the non-linearity function in the previous layers. The encoder embeds observations of the target dataset (shown in purple) and those from historical datasets (in green) into the resulting hidden representations, each of which is a r-dimensional vector. Subsequently, the attention unit attentively aggregates all these hidden representations into $\mathbf{r}_*$ which characterizes the underlying hyperparameter distribution. Note that the two layer MLP g first projects target configurations $\hat{\mathbf{x}}_j$ and historical configurations as the query and the value for attention, respectively. The attention unit, based on the basic multihead cross-attention (Vaswani et al., 2017), consists of 8 heads here. Specifically, $s^1, \cdots, s^M$, representing the similarity of the learned target dataset with the first and the M-th historical meta-dataset, modulate the dot products of keys and values. In the end, the decoder outputs the prediction $\hat{\boldsymbol{\mu}}_j$ and uncertainty $\hat{\boldsymbol{\sigma}}_j$ for a target configuration $\hat{\mathbf{x}}_j$.

A.2. The similarity for the dataset-aware attention

The similarity s^m between the target dataset and the m-th historical dataset, as introduced in Section 4.2, is calculated as,

$$s^m = \frac{1}{n_I + t} \sum_{t'} \cos(\mathbf{r}_{t'}, \frac{1}{Q} \sum_{t''}^{Q} \mathbf{r}^m_{t''}) = \frac{1}{n_I + t} \sum_{t'} \frac{\mathbf{r}_{t'} \cdot \frac{1}{Q} \sum_{t''} \mathbf{r}^m_{t''}}{\|\frac{1}{Q} \sum_{t''} \mathbf{r}^m_{t''}\| \|\mathbf{r}_{t'}\|}. \tag{1}$$

The intuition behind is that the learned hidden representations of observations characterize the latent distribution of hyperparameter performances, so that the similarity between datasets is conditioned on the hidden representations. For each $t'(1 \leq t' \leq n_I + t)$ configuration of the target dataset, we look up the Q-nearest neighbour configurations in the m-th dataset. If the mean hidden representation of these Q observations, i.e., $\frac{1}{Q}\sum_{t''}^{Q} \mathbf{r}^m_{t''}$, is very close to the hidden representation $\mathbf{r}_{t'}$ of the t'-th target observation, it implies that the performances of hyperparameter configurations of the two datasets bear striking similarity, at least locally in the t'-th configuration. By averaging over all $n_I + t$ observations in the target dataset, we obtain the similarity between the target dataset and the m-th dataset. Empirically, we have demonstrated $Q \in [2, 5]$ leads to a stable and superior similarity.

We also argue the advantages of the learned similarity over previous works. As mentioned in Section 2, a line of works exploits hand-crafted meta-features of a dataset to measure the similarity. Besides being hand-crafted, the resulting similarity is loosely connected to hyperparameter performances – it is likely that those hyperparameters outperforming on a previous dataset fail at the target, despite the high correlation between meta-features of the two datasets. More recently, Law et al. (Law et al., 2019) proposed to learn a neural network to obtain the embedding of a dataset and measure the similarity with the learned embedding. The major downside, obviously, is that the proposed fails to deal with datasets in heterogeneous feature spaces, e.g., the 100 OpenML datasets we investigate in this paper.

[1]Department of Computer Science, City University of Hong Kong, Hong Kong [2]Tencent AI Lab, Shenzhen, China. Correspondence to: Ying Wei <yingwei@cityu.edu.hk>.

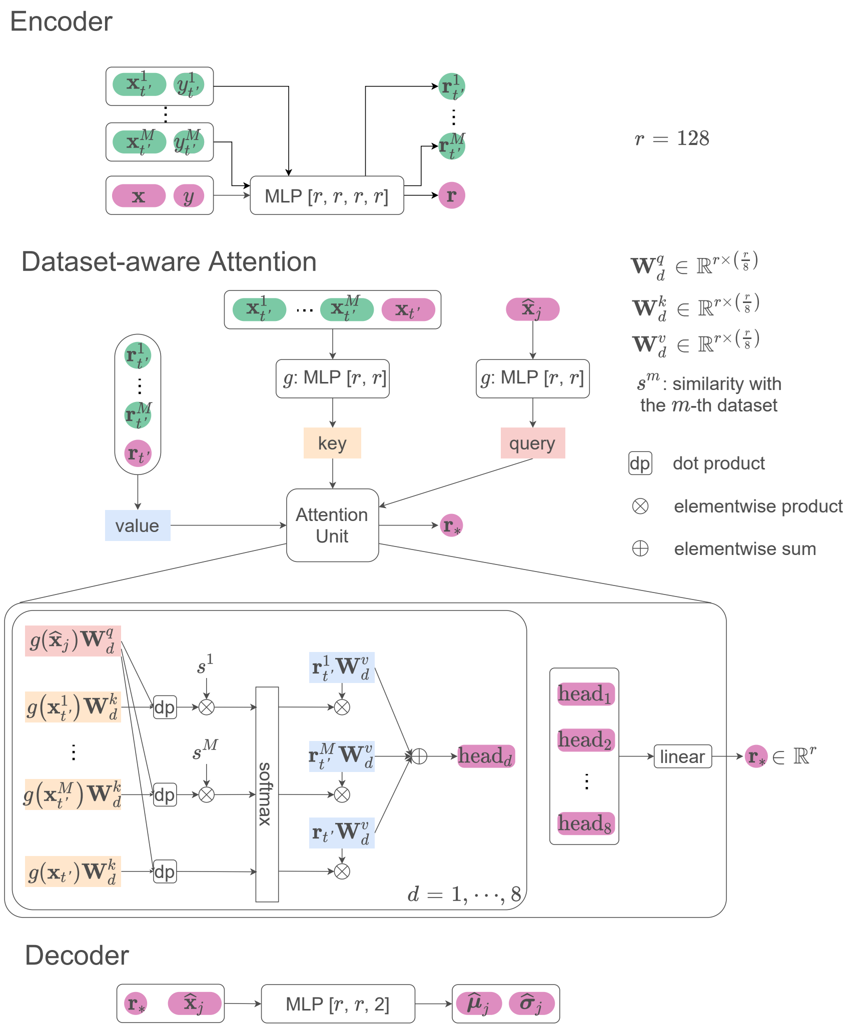


Figure 1: : The model architecture of the proposed TNP.

B. Baselines and Evaluation Metrics

B.1. Hyperparameter setting

First of all, we would claim that for all baselines we use the default hyperparameters for all datasets, just similar to our proposed TNP, since it is impractical to optimize the hyperparameters for every single dataset of 100 OpenML datasets.

As introduced in the experimental section, according to the surrogate model used, the baselines in our paper are categorized into three groups:

- **RS**: randomly search with no surrogate.
- **SMAC**: random forests are used as the surrogate.
- **GP, GP_LI, MTGP, EFFICIENT, RGPE**: the surrogate model in this case is Gaussian Processes. To deal with the hyperparameters of GP, e.g., those for the Matèrn-5/2 kernel, a widely accepted practice is to estimate them with MCMC sampling. The chain length and the number of burn-in steps for MCMC sampling are set to 200 and 100, respectively. Especially, for multitask GPs (**MTGP**) (Swersky et al., 2013), the kernel modelling dataset similarities is learned by simply encoding the dataset index. For **EFFICIENT** (Yogatama & Mann, 2014), the similarity between datasets is calculated based on meta-features. The kernel modelling the similarity between cross-dataset observations

is linearly combined with the kernel modelling the similarity between in-dataset observations, where the convex combination coefficient for the first kernel is set to 0.3.

- **DNGO, BOHAMIANN**: a fully connected neural network is taken as the surrogate model for these two baselines. The default network structure for **DNGO** and **BOHAMIANN** is a four-layer neural network with the number of hidden units being 50, i.e., $[50, 50, 50, 1]$. The tahn function is applied on the first three layers for non-linearity. Note that we have tried to increase the network capacity to align with TNP; nevertheless, the computational costs are prohibitively expensive, especially considering the huge number of OpenML datasets we work on. Moreover, the hyperparameter sensitive analysis in Figure 2 proves that TNP is not sensitive against the number of hidden units: it still outperforms even when the number of hidden units downgrades to 64. The default batch size for **DNGO** and **BOHAMIANN** is 10 and 20, respectively. The optimizer of Adam with a learning rate of 0.01 is used for **DNGO**, while the Momentum with a learning rate of 1e-5 and a decay rate of 0.05 is adopted in **BOHAMIANN**. By default, the hyperparameters for DNGO, i.e., the α and β for the Bayesian linear regression, are sampled via MCMC from the marginal log likelihood, where the chain length and the number of burn-in steps are 200 and 100, respectively. The similar applies to **BOHAMIANN** which directly samples network parameters via MCMC. We follow the implementation of the Bayesian neural network package[1].

B.2. Additional details for evaluation metrics

In this paper, we consider classification, so that we evaluate the performance of a HPO algorithm on a dataset via the best classification accuracy achieved so far at the t-th trial. However, to comprehensively evaluate an algorithm on all datasets, the classification accuracy is invalid as the classification difficulty varies from dataset and dataset – the mean classification accuracy over all datasets speaks nothing. As a result, we adopt the following metrics to eliminate the influence of the variation across datasets.

- **Average rank** At the t-th trial, each HPO algorithm gets a rank in terms of the best classification accuracy achieved so far. The best performing algorithm gets a rank of 1 and the worst algorithm gets the largest rank. By averaging the ranks of an algorithm over all datasets, we obtain the average rank of the algorithm along all trials.
- **Average distance to the maximum** At the t-th trial, the scaled distance between the classification accuracy achieved so far by an algorithm and the maximum accuracy across all algorithms and all trials is calculated for each algorithm, i.e.,
$$d_{t,a}^m = \frac{(\max_{t=1}^T \max_a y_{t,a}^m) - y_{t,a}^m}{(\max_{t=1}^T \max_a y_{t,a}^m) - (\min_{t=1}^T \min_a y_{t,a}^m)}, \tag{2}$$
where $y_{t,a}^m$ denotes the accuracy for the m-th dataset at the t-th trial by the algorithm a. By averaging the scaled distance over all datasets, i.e., $d_{t,a} = \sum_{m=1}^M d_{t,a}^m$, we obtain the resulting average distance to the maximum for the algorithm a at the t-th trial.
- **ECDF** We also investigate the empirical cumulative distribution of the number of datasets that have achieved the maximum accuracy at the t-th trial, where the maximum accuracy is again the highest among all trials and all algorithms. For example, if the value of ECDF at the t-th trial by an algorithm equals 0.3, it indicates that the algorithm achieved the maximum accuracy on 30% of all datasets at the t-th trial.

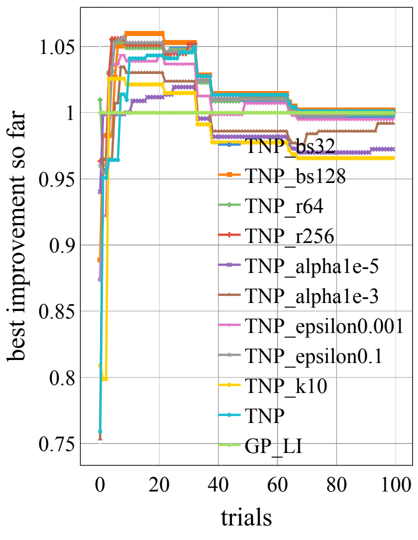
C. Experiments

C.1. Hyperparameter setting and sensitivity

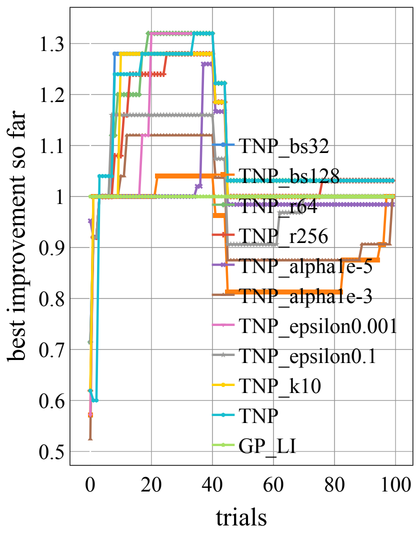
We summarize the hyperparameters in Table 1. TNP is implemented using Tensorflow (Abadi et al., 2016), and the codes will be released upon acceptance. Based on the following two observations, we conclude that TNP is not that sensitive against its hyperparameters.

- First, the fact that TNP with the default hyperparameters summarized in Table 1 consistently works well across all datasets sheds light on the insensitivity. As we mentioned above, we did not meticulously tune the hyperparameters, since tuning for each of the 100 OpenML datasets is impractical.

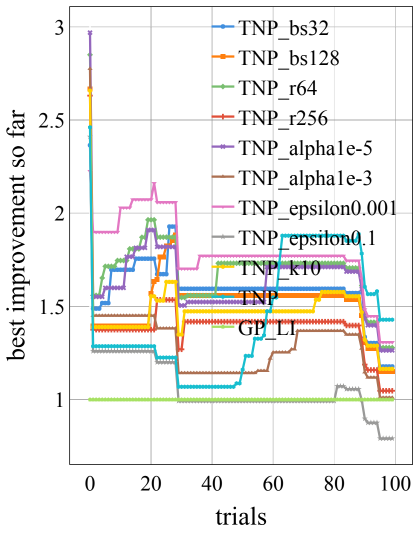
[1]https://github.com/automl/pybnn



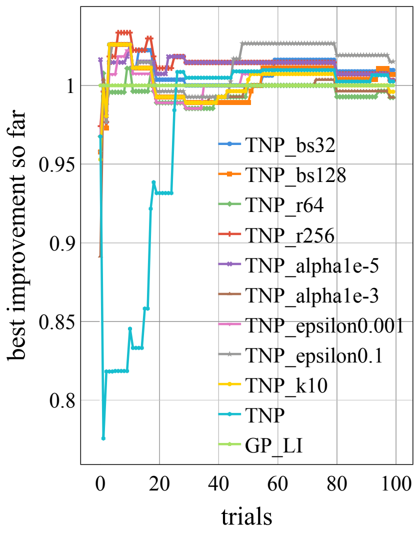
(a) : wine + oh5.wc → kr-vs-kp



(b) : fruitfly + BNG(lymph) → iris



(c) : new3s.wc + mammography → vowel



(d) : monks-problems-2 + tr12.wc → wap.wc

Figure 2: Comparing the improvement of all TNPs with different hyperparameters over GP_LI on four randomly sampled OpenML datasets.

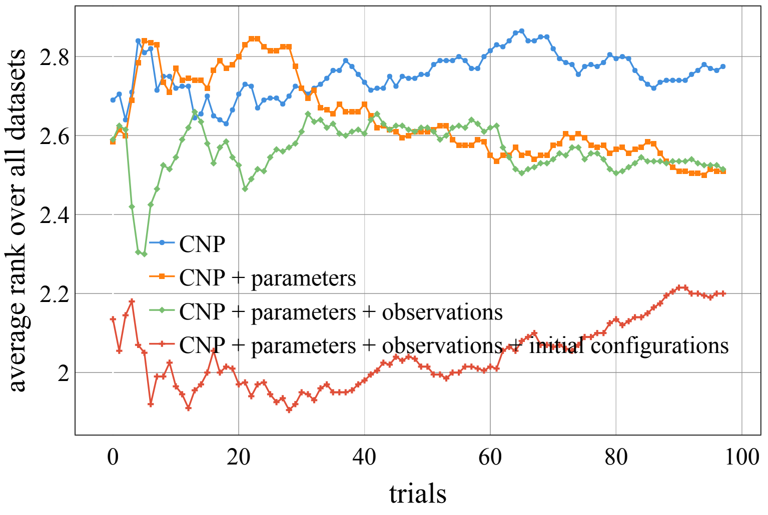
Table 1: Hyperparameter summary

Hyperparameters	Value
Number of hidden units (r)	128
Batch size (bs)	64
Learning rate (α)	0.0001
Meta update rate (ϵ)	0.01
Number of steps for inner learning (k)	5
Number of initial configurations (n_I)	3
Acquisition function (a)	EI
The number of neighbours for calculating dataset similarity (Q)	2

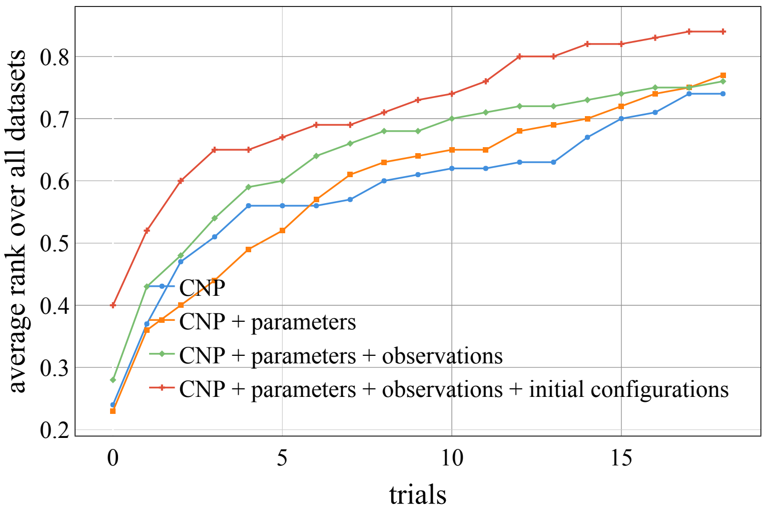
- Second, we have studied the hyperparameter sensitivity in Figure 3 in the main text, while here we also randomly select four datasets and demonstrate the hyperparameter sensitivity on them. Particularly, in Figure 2c and 2d, TNP with the default hyperparameters even does not rank high, compared to TNPs with the other hyperparameters.

C.2. Additional ablation studies

Here we extend the ablation study in Section 5.2 in the main text. First of all, we further investigate the four ablations of TNP investigated in Section 5.2, i.e., removing different components, in Figure 4 by comparing the ECDF. Again, the conclusions remain consistent with the comparison in terms of average rank in Section 5.2 – learning initial configurations is of particular significance in the beginning. Secondly, we also investigate the ablations by adding the components one by one to the vanilla conditional Neural Processes, as shown in Figure 3. Transferring parameters (CNP + parameters) mainly contributes to the fitting capability of neural processes as the surrogate after sufficient hyperparameter performances are observed, while leveraging observations (CNP + parameters + observations) also improves the early HPO process. Learning the initial configurations (CNP + parameters + observations + initial configurations) is the most contributory to warm-start HPO. The results strongly advocate that all the components are contributory and even mutually reinforcing, and further corroborate our contribution, i.e., empowering the simultaneous transfer of parameters, initial configurations, and



(a) : Average rank



(b) : ECDF

Figure 3: Comparison of all four ablations of TNP in terms of (a) average rank and (b) ECDF.

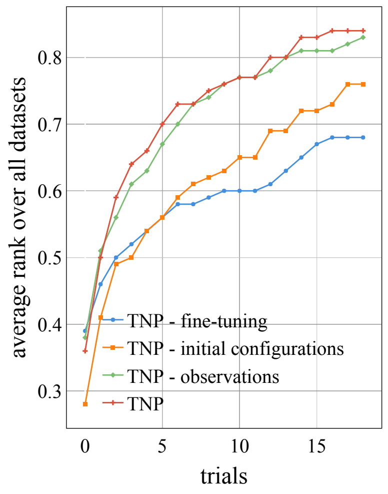


Figure 4: Comparison of four ablations of TNP in terms of ECDF.

observations via Neural Processes.

C.3. Additional results for varying the number of meta-datasets

In the experimental section, we have demonstrated that when the number of meta-datasets increases to 50, TNP is still qualified to transfer the knowledge from similar tasks and outperforms other transfer baselines. Here we would like to present more results of varying the number of meta-datasets.

- We compare all transfer algorithms that transfer knowledge from previous datasets by varying the number of meta datasets to be $M = 2$, $M = 10$ and $M = 20$ in Figure 5a, 5b, and 5c, respectively. Note that the comparison result on $M = 50$ has been reported in the main text. When the number of meta-datasets increases, some algorithms such as RGPE tend to deteriorate. The reason possibly lies in that more meta-datasets also bring more noisy or irrelevant datasets, which requires more accurate estimation of the similarity between datasets. However, TNP learns a robust similarity between datasets and consistently outperforms the other transfer baselines regardless of the number of meta-datasets.
- In Figure 5d, we also compare the TNPs with different numbers of meta-datasets. Generally speaking, as the number of meta-datasets increases, TNP is capable of effectively transferring more knowledge from more meta-datasets, thereby improving HPO for the target dataset. This empirical result lays the foundation for implementation of TNP in real-world applications where more and more historical datasets are accessible over time.
- We also randomly sample four datasets from OpenML and compare the best classification accuracies achieved by TNPs when different numbers of meta-datasets are leveraged . Figure 6 further supports the conclusion that TNP is qualified to harness the diversified knowledge contributed by more meta-datasets.

C.4. Results for varying the number of initialization configurations

In Figure 7, we comprehensively investigate the influence of the number of initial configurations.

- We compare all algorithms in terms of the average rank by varying the number of initial configurations to be $n_I = 1$, $n_I = 5$, and $n_I = 10$ in Figure 7a, Figure 7b, and Figure 7c, respectively. Note that we do not report the results on $n_I = 3$, which is the default setting and has been extensively studied and reported in the main text. Besides, we do not consider too many initial configurations, i.e., $n_I > 10$, since the objective of TNP is exactly to expedite HPO in as few as trials as possible and get rid of the computationally exhaustive evaluation of a configuration required in a trial. When $n_I = 1$, the algorithms without learned initial configurations (e.g., GP) are almost as competitive as those that learn initial configurations (e.g., RGPE_LI) at the beginning of HPO. It is challenging to learn an effective

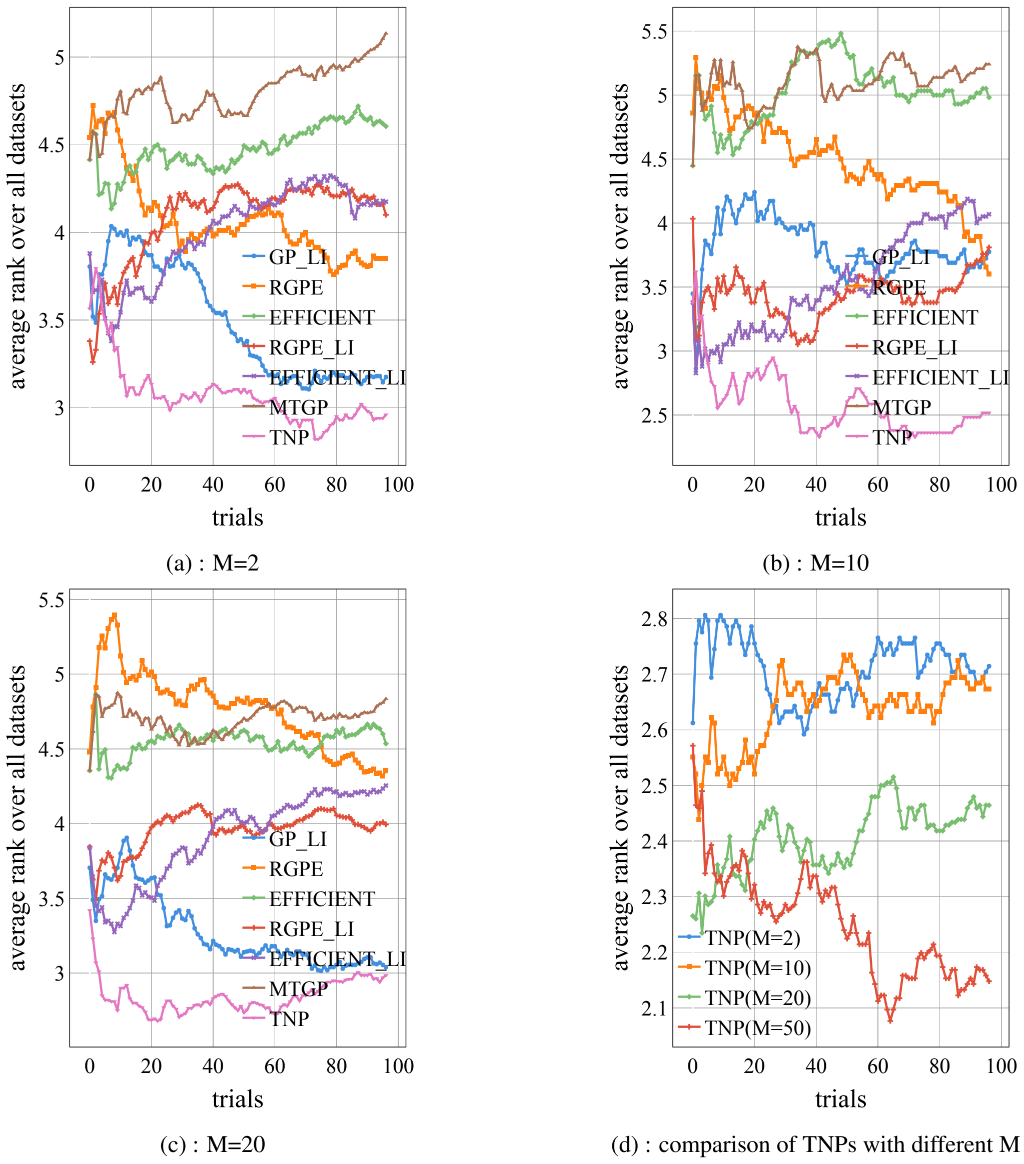


Figure 5: Comparing the average rank of all transfer baselines as well as TNP by varying the number of meta-datasets, e.g., (a) M=2, (b) M=10, and (c) M=20. (d) compares the TNPs with different numbers of meta-datasets.

initial configuration in a single trial. However, when n_I increases to 5, the algorithms learning initial configurations (e.g., RGPE_LI) greatly outperform the others, especially in the first few trials. Further, more initial configurations with $n_I = 10$ again reduce the difference between the algorithms with the initial configurations learned and those without – the random initial configurations in sufficient trials (i.e., 10) have a higher chance to be as good as the learned configurations. In either case, TNP demonstrates its robustness and superiority over other baselines.

- Figure 7d compares the TNPs with different numbers of initial configurations. As expected, more initial configurations contribute better HPO performances. TNP with $n_I = 1$ is the worst, as single observation hinders the reliability of the learned similarity between datasets and impairs the effectiveness of knowledge transfer. While considering the additional evaluation costs taken by more initial configurations, we argue that $n_I = 3$ or $n_I = 5$ is enough for TNP.

C.5. Additional results for the influence of random seeds

Based on the following two perspectives, we conclude that TNP is robust against random seeds.

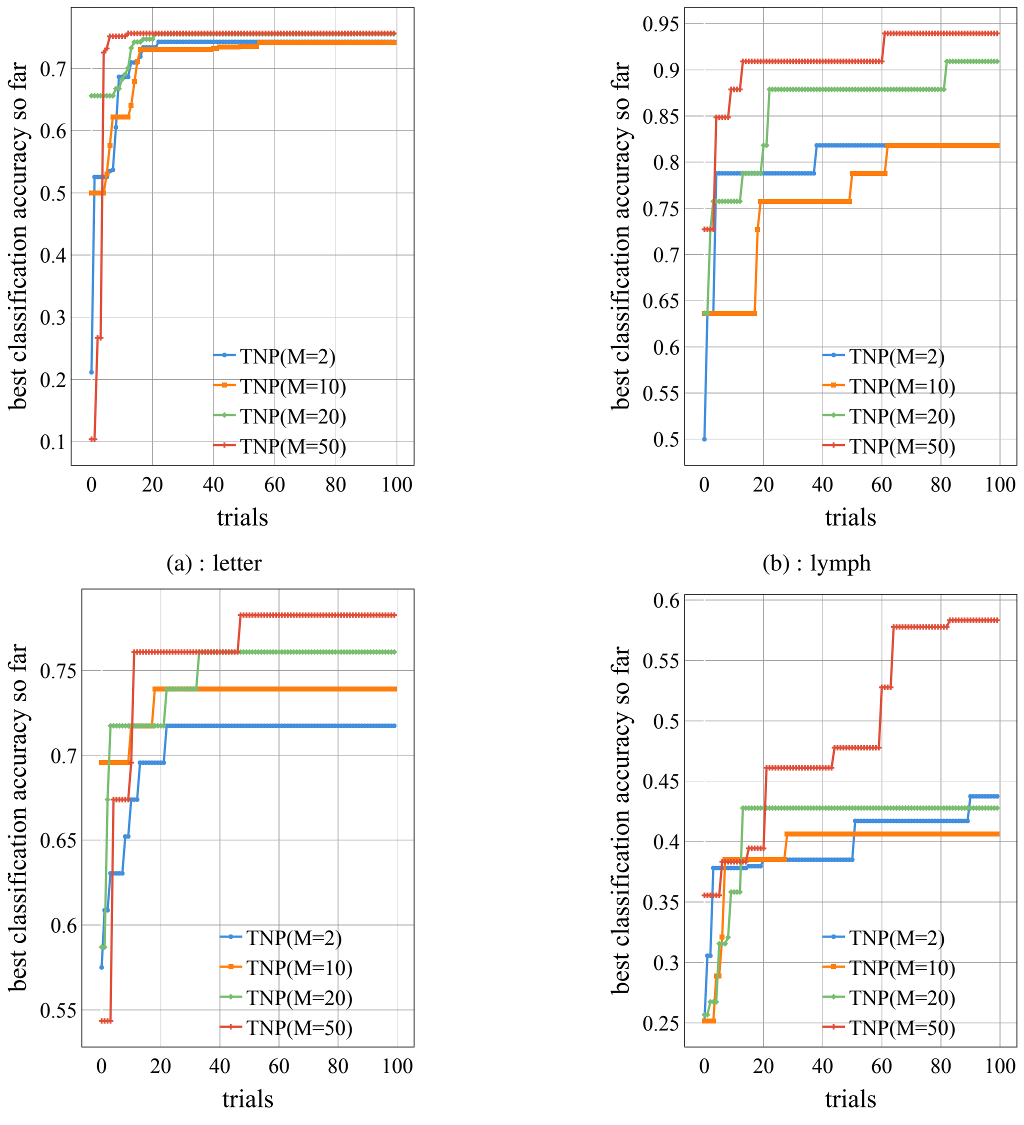


Figure 6: Further investigating the influence of the number of meta-datasets via four randomly sampled OpenML datasets, in terms of the best classification accuracy achieved so far.

- For each dataset, we repeatedly perform all algorithms 10 times under 10 different random seeds. For fair comparison, we apply the same seed to all algorithms each time, both for initial configuration and dataset splitting, so that the influence of the seed on different algorithms is eliminated. Besides Figure 4 and Figure 8 in the experimental section, here we present more classification accuracy comparison on another four randomly sampled datasets from OpenML in Figure 8. Remarkably, TNP consistently outperforms with a comparative small variance.
- More importantly, the average performance over all 100 OpenML datasets shown in Figure 3 in the experimental section in the main text is actually averaged over 100×10 different seeds – the superiority of TNP consolidates the robustness.

C.6. Additional results for comparison of CPU runtime overhead

In Figure 9, we further investigate the CPU time overhead taken by different algorithms along trials, when the number of meta-datasets varies. TNP is almost as efficient as GP and GP_LI, both of which yet do not transfer observations from past

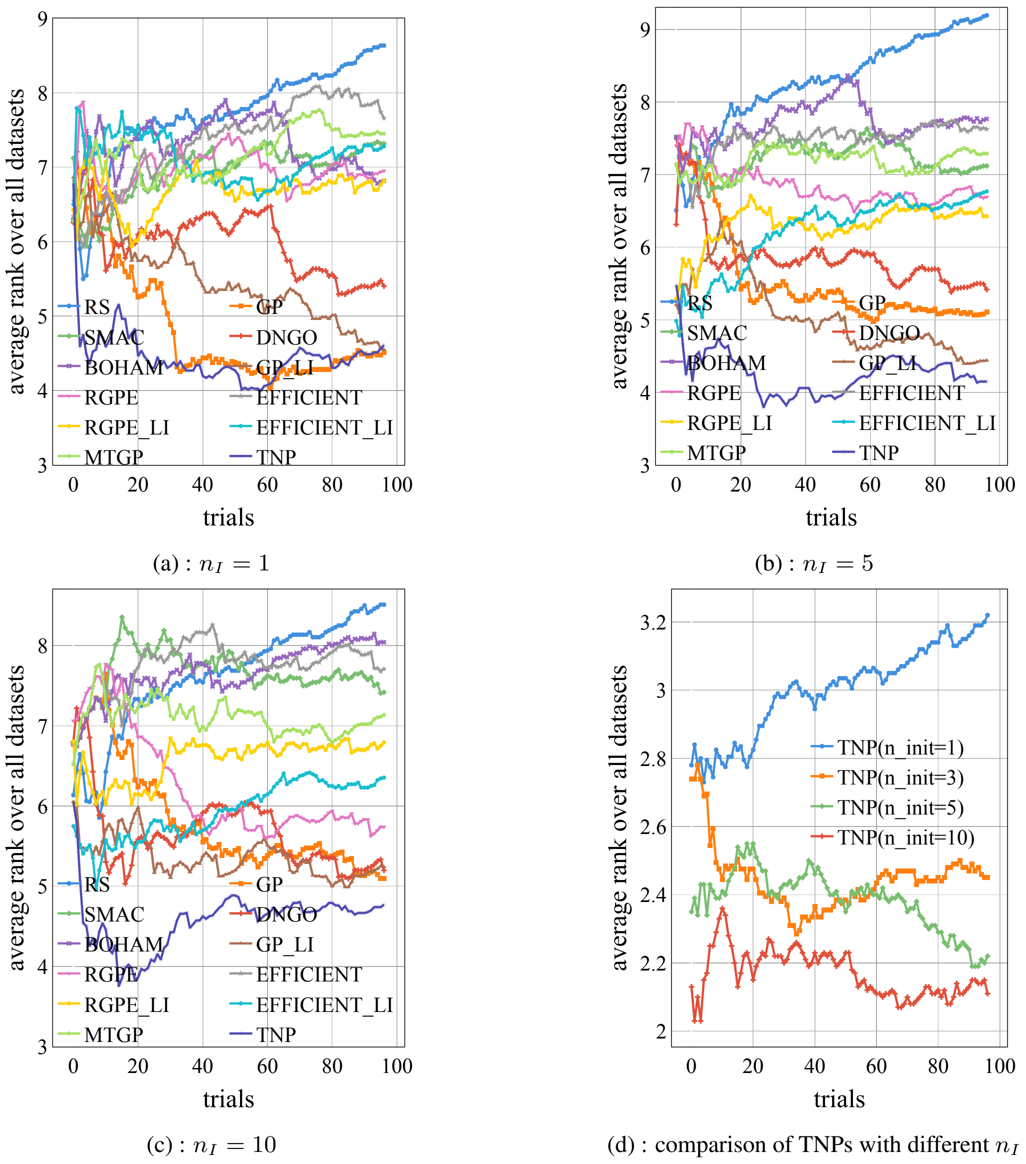
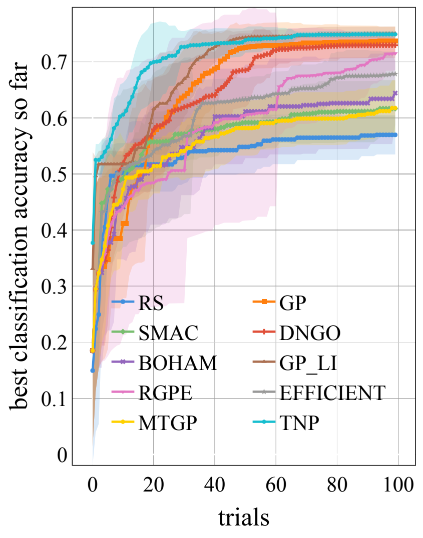


Figure 7: Comparing the average rank of all baselines well as TNP by varying the number of initial configurations, e.g., (a) $n_I = 1$, (b) $n_I = 5$, and (c) $n_I = 10$. (d) compares the TNPs with different numbers of initial configurations.

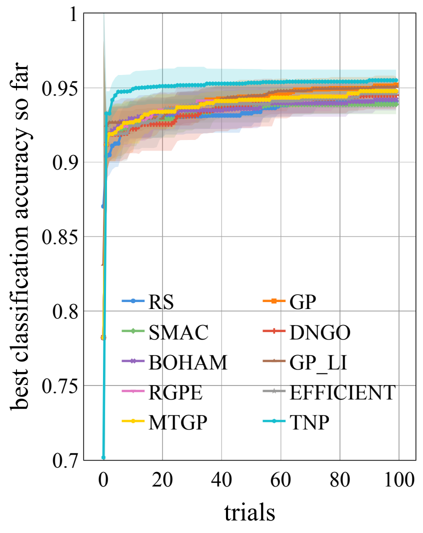
HPO experiences. In contrast, MTGP and EFFICIENT that leverage HPO observations on other datasets are computationally prohibitive, even at the beginning of HPO. RGPE which transfers GP models does not suffer the intensive computational cost at the beginning of HPO, while it scales almost exponentially as the number of trials increases. It is because the observations (say 90 observations at the 90-th trial) are expected to be considered by all M transferred GP models. We highlight that TNP is an effecitve HPO algorithm by transferring the joint of observations, parameters, initial configurations, but meanwhile remains as efficient as non-transfer baselines such as GP.

C.7. Effectiveness of the learned similarity

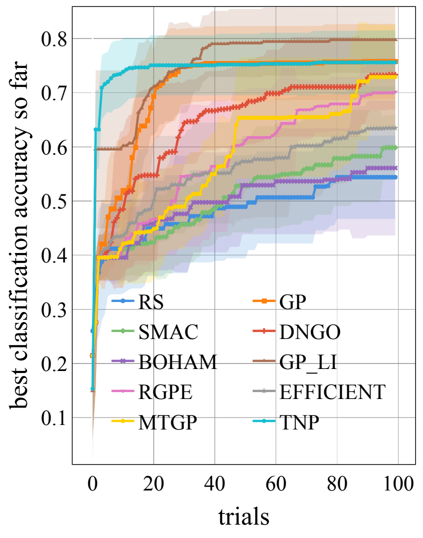
As detailed above in A.2, the learned similarity is based on hidden representations of all observations in a dataset, which is not only free from manually defined meta-features but also more descriptive and pertinent to HPO behaviours. Here we verify the superiority of the learned similarity over that computed according to hand-crafted meta-features, by replacing the similarity in the dataset-aware attention unit with the similarity calculated with meta-features. Figure 10a demonstrates



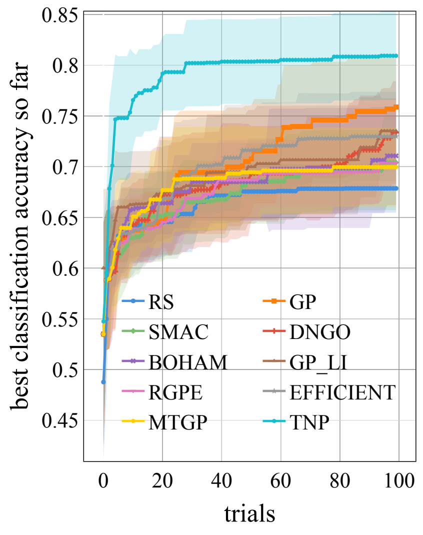
(a) : mfeat-pixel + waveform-5000 → letter



(b) : monks-problems-3 + pol → page-blocks



(c) : ailerons + mfeat-fourier → ecoli



(d) : splice + abalone → spambase

Figure 8: Comparison of the best accuracies achieved so far on four additional randomly sampled OpenML datasets, where each algorithm is performed 10 times.

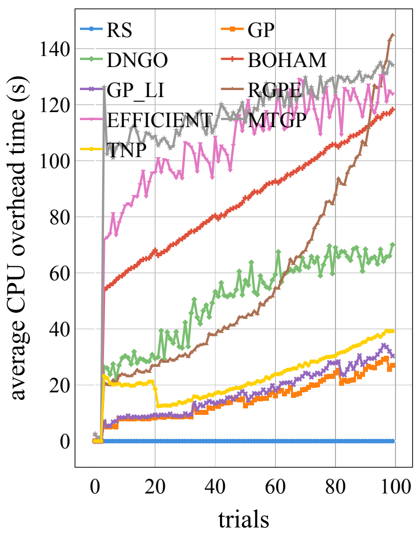
the effectiveness of the learned similarity, in terms of the average rank over all 100 OpenML datasets. We also show the superiority of the learned similarity over the similarity calculated with meta-features on two randomly sampled datasets in Figure 10b and 10c.

References

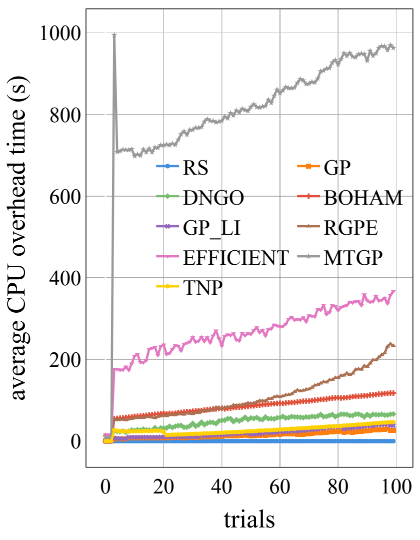
Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., Devin, M., Ghemawat, S., Irving, G., Isard, M., et al. Tensorflow: A system for large-scale machine learning. In *OSDI*, pp. 265–283, 2016.

Kim, H., Mnih, A., Schwarz, J., Garnelo, M., Eslami, A., Rosenbaum, D., Vinyals, O., and Whye Teh, Y. Attentive neural processes. In *ICLR*, 2019.

Law, H. C., Zhao, P., Chan, L. S., Huang, J., and Sejdinovic, D. Hyperparameter learning via distributional transfer. In *NIPS*, pp. 6801–6812, 2019.



(a) : M=10



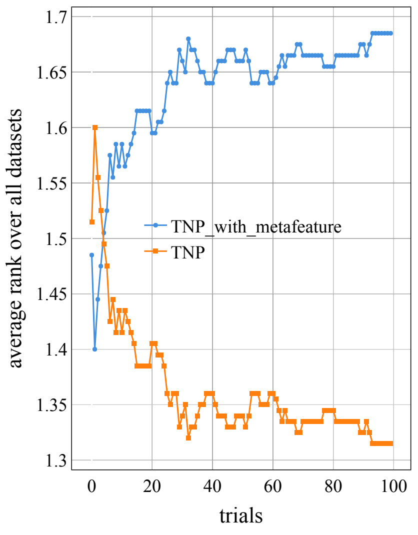
(b) : M=20

Figure 9: Comparing the CPU runtime overhead of all baselines as well as TNP, when the number of meta-datasets (a) M=10 and (b) M=20.

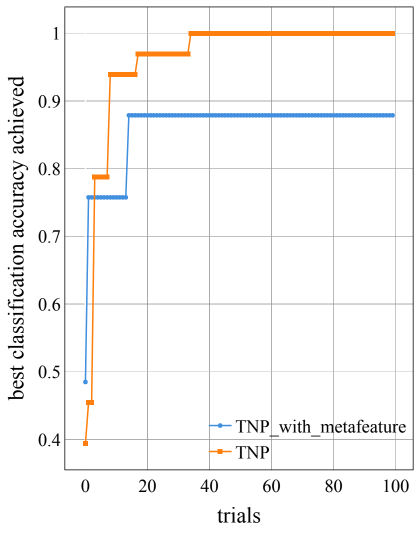
Swersky, K., Snoek, J., and Adams, R. P. Multi-task bayesian optimization. In *NIPS*, pp. 2004–2012, 2013.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. Attention is all you need. In *NIPS*, pp. 5998–6008, 2017.

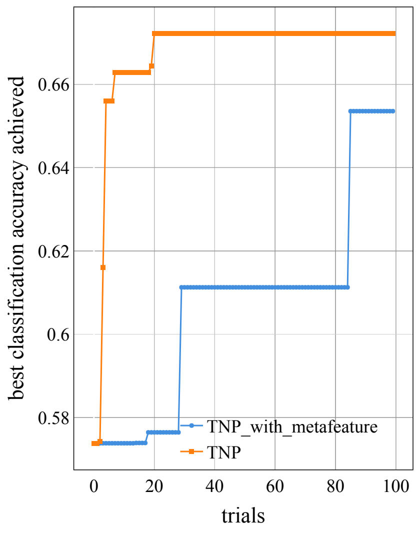
Yogatama, D. and Mann, G. Efficient transfer learning method for automatic hyperparameter tuning. In *AISTATS*, pp. 1077–1085, 2014.



(a) : Average rank



(b) : fruitfly + BNG(lymph) $\rightarrow$ iris



(c) : pwLinear + monks-problems-1 $\rightarrow$ electricity

Figure 10: The average ranks of our TNP and the TNP with the dataset similarity calculated from hand-crafted meta-features, as well as the best classification accuracies achieved by them on two randomly sampled OpenML datasets.