Transferable Neural Processes for Hyperparameter Optimization

Ying Wei[†], **Peilin Zhao**[†], **Huaxiu Yao**[‡], **Junzhou Huang**[†]
[†]Tencent AI Lab, Shenzhen, China
[‡]Pennsylvania State University, State College, PA 16801, USA
judyweiying@gmail.com, masonzhao@tencent.com, huaxiuyao@psu.edu, jzhuang@uta.edu

Abstract

Though conventional hyperparameter optimization (HPO) algorithms work well when abundant trials are available, they are far from satisfactory in practical time-demanding tasks where an optimal hyperparameter configuration is expected to return in as few trials as possible. In this paper, we propose an end-to-end algorithm called Transfer Neural Processes (TNP) to speed up HPO, by simultaneously transferring three types of knowledge, i.e., observations, parameters, and initial configurations, from historical HPO trials on other datasets. The proposed TNP achieves state-of-the-art performance in at least one order of magnitude less trials.

1 Introduction

Sequential model-based optimization (SMBO) [10] has been the current state-of-the-art for HPO. The core of SMBO is to learn from observed hyperparameter performances a surrogate model which maps a hyperparamter configuration to the evaluation measure on a dataset. Sequentially, in each trial, a promising configuration estimated by the surrogate is evaluated and this new observation is incorporated to further improve the surrogate. While existing surrogate models including Gaussian Processes (GPs) [17], parzen estimators [2], random forest [9], and neural networks [18, 19] have shown their effectiveness provided with sufficient observations, it is imperative to return an optimal configuration in very few trials in real-world applications where a trial on huge datasets is costly.

Here we are devoted to speed up HPO by transferring knowledge from historical trials on other datasets – a subset of hyperparameter configurations that perform well on similar datasets are likely qualified candidates for the target dataset. As the convention of transfer learning [14], there exist three research problems, i.e., when, what, and how to transfer. First, when to transfer here is grounded on measuring the similarity between datasets. Most of existing studies [1, 4, 16, 26] rely on meta-features of a dataset which, however, are hand-crafted and loosely related to hyperparameter performances. Second, despite the instantiation of what to transfer as either initializations [4, 13, 23], observations [16, 20, 26], parameters of a surrogate model [1, 3, 15, 25], or acquisition functions [24], none of existing works is qualified to harness the collective power of them. Third, in terms of how to transfer, almost all previous works develop GP-based surrogate models with cubic scaling which are highly inefficient to incorporate abundant past observations. One rencet work [15] applies Bayesian linear regression as GP approximation, but meanwhile it loses predictive power. Especially, they all require an explicitly defined kernel, e.g., the linear kernel in [15] which is fixed across datasets, being inadequate to accomodate a heterogeneous dataset in practical scenarios.

To address these problems, we propose a novel end-to-end algorithm called Transfer Neural Processes (TNP). Motivated by recent success of Neural Processes (NPs) [6], we adopt NPs as our surrogate model. By combining the best of both GPs and neural networks, NPs preserve the property reminiscent of GP, i.e., defining distributions over functions, and meanwhile be efficiently trained with standard deep learning libraries. The TNP consists of an encoder, a dataset-aware attention unit, and a decoder. The model achieves transfer learning by simultaneously leveraging observations of previous

33rd Conference on Neural Information Processing Systems (NeurIPS 2019), Vancouver, Canada.

datasets, learning from all datasets a transferable initialization for parameters of the TNP, as well as optimizing a well-generalized initial set of configurations to evaluate for SMBO. The dataset-aware attention unit evaluates the similarity between datasets using the encoded representations of all observations in a dataset, thereby eliminating the need of manually defining meta-features. Moreover, the TNP modelling an implicit kernel is fine-tuned with several gradient updates for a target dataset, which empowers TNP to meet more wildly heterogeneous datasets.

2 Background and Problem Setup

Given a dataset $\mathcal{D} \sim \mathcal{P}_D$, *hyperparameter optimization (HPO)* aims to identify optimal values for hyperparameters $\mathbf{x}$ so that the value of a hyperparameter response function f is maximized (or minimized), i.e., $\mathbf{x}^* = \arg\max_{\mathbf{x}\in\mathcal{X}} f(\mathbf{x})$. As a dominant framework for HPO, *Sequential Model-based Bayesian Optimizaion (SMBO)* [10] starts by querying the function f at n_I initial configurations $\mathbf{x}_{I0}, \cdots, \mathbf{x}_{In_I}$ to constitute the initial set of history observations $\mathcal{H}_0 = \{(\mathbf{x}_{I0}, y_{I0}), \cdots, (\mathbf{x}_{In_I}, y_{In_I})\}$. Afterwards, it iterates the following four stages: 1) in the t-th trial, fit the surrogate Φ_t on the observations $\mathcal{H}_{t-1}$; 2) use the surrogate Φ_t to make predictions $\{\hat{\mu}_j\}_{j=0}^{n_\mathcal{X}}$ with uncertainties $\{\hat{\sigma}_j\}_{j=0}^{n_\mathcal{X}}$ for $n_\mathcal{X}$ target configurations $\{\hat{\mathbf{x}}_j\}_{j=0}^{n_\mathcal{X}}$; 3) based on the predictions and uncertainties, the acquisition function a decides the next configuration $\mathbf{x}_t \in \{\hat{\mathbf{x}}_j\}_{j=0}^{n_\mathcal{X}}$ to try; 4) evaluate the function f at $\mathbf{x}_t$, and update the history set $\mathcal{H}_t = \mathcal{H}_{t-1} \cup \{(\mathbf{x}_t, y_t)\}$. In the t-th trial, there are a total number of $n_I + t$ observations in the history set $\mathcal{H}_{t-1}$. In this paper, additionally, we leverage knowledge from M history sets, i.e., $\mathcal{H}^1_{T^1}, \cdots, \mathcal{H}^M_{T^M}$, on M datasets, i.e., $\mathcal{D}^1, \cdots, \mathcal{D}^M$. In the m-th dataset, there are T^m observations available, i.e., $\mathcal{H}^m_{T^m} = \{(\mathbf{x}^m_t, y^m_t)\}_{t=1}^{T^m}$. The goal of this paper lies that by borrowing strength from these M history sets, the surrogate can be quickly maximized (equivalent to maximizing the response function f) with the optimal hyperparameter configuration returned in less trials.

3 Transferable Neural Processes

The neural process model, based on NPs [6, 7, 11], involves three components. The encoder learns an embedding $\mathbf{r}_{t'} \in \mathbb{R}^r$ for each observation $(\mathbf{x}_{t'}, y_{t'})$, i.e., $\mathbf{r}_{t'} = E_{\theta_e}(\mathbf{x}'_t, y'_t), \forall t' \in \{0, \cdots, n_I + t\}$. The encoder E_{θ_e} is parameterized with a neural network. The dataset-aware attention unit as the second component summarizes all observations and produces an order-invariant representation of historical observations. This representation, $\mathbf{r}_* = A_{\theta_a}(\mathbf{r}_0, \cdots, \mathbf{r}_{n_I+t}) \in \mathbb{R}^r$, is expected to encode the latent distribution of hyperparameter performances conditioned on the whole set of observations $\mathcal{H}_{t-1}$. Last, the decoder takes the representation $\mathbf{r}_*$ as well as a target configuration $\hat{\mathbf{x}}_j$ as input, and outputs $\hat{\mathbf{y}}_j \in \mathbb{R}^2$ as predictions on values of f, i.e., $\hat{\mathbf{y}}_j = D_{\theta_d}(\mathbf{r}_*, \hat{\mathbf{x}}_j)$. The two values of $\hat{\mathbf{y}}_j$ represent the mean $\hat{\mu}_j$ and variance $\hat{\sigma}_j$ of a Gaussian distribution $\mathcal{N}(\hat{\mu}_j, \hat{\sigma}_j)$, respectively. We also parameterize the decoder D_{θ_d} with a neural network. Draw inspiration from [7], we train the parameters, i.e., $\theta = \theta_e \cup \theta_a \cup \theta_d$, by following three steps: 1) randomly shuffle observations in $\mathcal{H}_{t-1}$ and divide them into two parts, e.g., $\mathcal{H}_{t-1,h} = \{(\mathbf{x}_{t'}, y_{t'})\}_{t'=0}^{t_h}$ and $\mathcal{H}_{t-1,\bar{h}} = \{(\mathbf{x}_{t'}, y_{t'})\}_{t'=t_h+1}^{n_I+t}$; 2) predict the observations $\mathcal{H}_{t-1,h}$ conditioned on $\mathcal{H}_{t-1,\bar{h}}$; 3) maximize the conditional log likelihood,

$$\mathcal{L}(\mathcal{H}_{t-1,h}, \mathcal{H}_{t-1,\bar{h}}|\theta) = \mathbb{E}_{f\sim P}[\mathbb{E}_{t_h}[\log p_\theta(\{y_{t'}\}_{t'=0}^{t_h}|\mathcal{H}_{t-1,\bar{h}}, \{\mathbf{x}_{t'}\}_{t'=0}^{t_h})]], \tag{1}$$

where the gradient of the loss is practically estimated by sampling f from an underlying distribution P and sampling different values of t_h. The pictorial overview of TNP is shown in Appendix.

Transferring observations via dataset-aware attention The crux of GPs lies in modelling the similarity between a target configuration and configurations of past observations, while leveraging observations from other datasets requires another desiderata, i.e., similarity between datasets. With multihead attention [22], we design our dataset-aware attention A_{θ_a} as $\mathbf{r}_* = A_{\theta_a}(\mathbf{r}_0, \cdots, \mathbf{r}_{n_I+t}, \mathbf{r}^1_0, \cdots, \mathbf{r}^M_{T^M}) = \textbf{MultiHead}(g(\hat{\mathbf{x}}_j), g(\mathbf{X}^{0:M}), \mathbf{R}^{0:M}, \mathbf{s})$ where $g(\hat{\mathbf{x}}_j) \in \mathbb{R}^r$ is the query, and $g(\mathbf{X}^{0:M})$ serves as the keys with $\mathbf{X}^{0:M} = [\mathbf{X}; \mathbf{X}^1; \cdots; \mathbf{X}^M]$ including both in-dataset observations $\mathbf{X}$ and cross-dataset ones $\mathbf{X}^m$. $\mathbf{R}^{0:M}$ provides the values to be attentively aggregated. The final $\mathbf{r}_*$ concatenates all $\lfloor r/h \rfloor$ heads, with each $\text{head}_h := \text{softmax}\big(\mathbf{s} \circ [g(\hat{\mathbf{x}}_j)\mathbf{W}^q_h][g(\mathbf{X}^{0:M})\mathbf{W}^k_h]^T/\sqrt{r}\big)\mathbf{R}^{0:M}\mathbf{W}^v_h$. Note that $\mathbf{W}^q_h$, $\mathbf{W}^k_h$, $\mathbf{W}^v_h \in \mathbb{R}^{r\times h}$ are parameters. We especially highlight $\mathbf{s} = \text{softmax}([1, s^1\mathbf{1}^{(1\times T^1)}, \cdots, s^M\mathbf{1}^{(1\times T^M)}])$ which measures the similarity between the target and all datasets. The similarity is estimated as $s^m = (\frac{1}{T^m}\sum_{t'}\mathbf{r}^m_{t'} \cdot \frac{1}{n_I+t}\sum_{t'}\mathbf{r}_{t'})/(\|\frac{1}{T^m}\sum_{t'}\mathbf{r}^m_{t'}\|\|\frac{1}{n_I+t}\sum_{t'}\mathbf{r}_{t'}\|)$, where we condition on the mean of embeddings of all observations in a dataset. Besides liberating practitioners from manually defining meta-features of a dataset, the mean is more descriptive and pertinent to the HPO behaviours.

Algorithm 1: Transferable Neural Processes (TNP) for Hyperparameter Optimization

Input : Observations on M datasets $\mathcal{H}^1_{T^1}, \cdots, \mathcal{H}^M_{T^M}$; # of trials T; acquisition function a; target configurations $\{\hat{\mathbf{x}}_j\}_{j=0}^{n_{\mathcal{X}}}$; meta update rate ϵ; # of initial configurations n_I.

Output : The best hyperparameter configuration $\mathbf{x}^*$ found.

Randomly initialize $\tilde{\theta}$, $\{\tilde{\mathbf{x}}_{Ij}\}_{j=1}^{n_I}$, and set $y^* \leftarrow \infty$;

for $m = 1, \cdots, M$ **do** /* Meta-training */

  Perform k gradient steps on : $\theta_k^m = \tilde{\theta} - \alpha \nabla_\theta^k \mathcal{L}(\mathcal{H}^m_{T^m,h}, \mathcal{H}^m_{T^m,\bar{h}}|\theta)$;

  $\mathbf{x}_{Ij}^k = \tilde{\mathbf{x}}_{Ij} - \alpha \nabla_{\mathbf{x}_{Ij}}^k \mathcal{L}_I(\{\mathbf{x}_{Ij}\}_{j=1}^{n_I}|\theta), \forall j = 0, \cdots, n_I$;

  Update $\tilde{\theta}$ and $\{\tilde{\mathbf{x}}_{Ij}\}_{j=1}^{n_I}$: $\tilde{\theta} = \tilde{\theta} + \epsilon(\theta_k^m - \tilde{\theta})$, $\tilde{\mathbf{x}}_{Ij} = \tilde{\mathbf{x}}_{Ij} + \epsilon(\mathbf{x}_{Ij}^k - \tilde{\mathbf{x}}_{Ij})$;

end

Query the values of f at $\{\tilde{\mathbf{x}}_{Ij}\}_{j=1}^{n_I}$, and obtain the initial observation set $\mathcal{H}_0 = \{(\tilde{\mathbf{x}}_{Ij}, \tilde{y}_{Ij})\}_{j=0}^{n_I}$;

for $t = 1, \cdots, T$ **do** /* Meta-test */

  Fine-tune TNP by k gradient steps: $\theta_k = \tilde{\theta} - \alpha \nabla_\theta^k \mathcal{L}(\mathcal{H}_{t-1,h}, \mathcal{H}_{t-1,\bar{h}}|\theta)$;

  Fit TNP_{θ_k} to $\mathcal{H}_{t-1}$: $\mathbf{x}_t \leftarrow \arg\max_{\mathbf{x} \in \{\hat{\mathbf{x}}_j\}_{j=0}^{n_{\mathcal{X}}}} a(\text{TNP}_{\theta_k}(\mathbf{x}))$;

  Evaluate $y_t = f(\mathbf{x}_t)$ and update the observation set $\mathcal{H}_t = \mathcal{H}_{t-1} \cup \{(\mathbf{x}_t, y_t)\}$;

  if $y_t > y^*$ **then**

    $\mathbf{x}^*, y^* \leftarrow \mathbf{x}_t, y_t$;

  end

end

return $\mathbf{x}^*$;

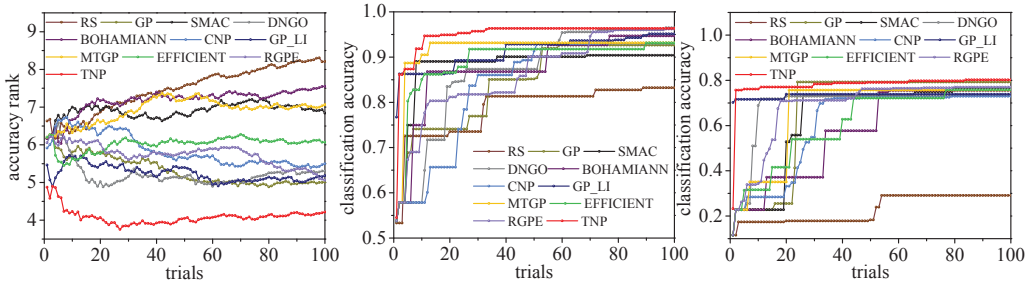
Transferring parameters Under the assumption that different response functions f sampled to optimize Eqn. (1) by sampling datasets are from the same distribution, the implicit kernel modelled by TNP is globally shared and progressively improved as training proceeds across datasets. Unfortunately, this assumption runs counter to practical scenarios where a global kernel cannot accommodate diverse datasets. To alleviate the problem, we follow the strategy in model agnostic meta-learning [5] which has been proved its equivalence to hierarchical Bayesian inference [8]. Specifically, in meta-training, each time we sample the m-th of of M datasets as the target and the rest as historical datasets. First, initialized with the globally shared parameters $\tilde{\theta}$, TNP optimizes in k gradient steps to obtain dataset-specific parameters θ_k^m, i.e., $\theta_k^m = \tilde{\theta} - \alpha \nabla_\theta^k \mathcal{L}(\mathcal{H}^m_{T^m,h}, \mathcal{H}^m_{T^m,\bar{h}}|\theta)$, where we follow Eqn. (1) by dividing $\mathcal{H}^m_{T^m}$ into two parts. In turn, θ_k^m updates the transferable initialization $\tilde{\theta}$ with $\tilde{\theta} = \tilde{\theta} + \epsilon(\theta_k^m - \tilde{\theta})$. During meta-test, it is straightforward to first fine-tune TNP on $\mathcal{H}_{t-1}$, i.e., $\theta_k = \tilde{\theta} - \alpha \nabla_\theta^k \mathcal{L}(\mathcal{H}_{t-1,h}, \mathcal{H}_{t-1,\bar{h}}|\theta)$, and then make predictions for a target configuration using the TNP parameterized with θ_k.

Initializing SMBO with well-generalized configurations The initial configurations have been demonstrated crucial to the success of SMBO [13, 23] – those configurations which achieve larger values of f are prone to speed up SMBO. The M observation sets $\mathcal{H}^1_{T^1}, \cdots, \mathcal{H}^M_{T^M}$ offer a treasure of the configurations with higher f values. Therefore, we again formulate the problem of learning initial configurations as a hierarchical Bayesian inference problem. Similar to inferring $\tilde{\theta}$, we learn the set of well-generalized initial configurations $\{\tilde{\mathbf{x}}_{Ij}\}_{j=1}^{n_I}$ which are fine-tuned for each m-th dataset. The only difference is the loss with regards to $\{\tilde{\mathbf{x}}_{Ij}\}_{j=1}^{n_I}$, which enforces that the predictions of at least one of the initial configurations are maximized, i.e., $\mathcal{L}_I(\{\mathbf{x}_{Ij}\}_{j=1}^{n_I}|\theta) = \sum_{j=1}^{n_I}(e^{\alpha\mu_{Ij}})\mu_{Ij} / (\sum_{j'}^{n_I} e^{\alpha\mu_{Ij'}})$.

4 Experiments

The encoder, the decoder, and the attention embedding function g are all implemented as a two layer multilayer perceptron with $[128, 128]$ hidden units, which indicates $r = 128$. Following [6, 11], we first pre-train the networks by sampling $30,000$ batches of $n_{\mathcal{X}}$ dimensional GP functions with the length scale $l \sim U[0.3, 1.0]$ and the kernel scale $\sigma = 1.0$. Note that we set the batch size, the number of gradient steps k, and the learning rate α for Adam, and the meta update rate ϵ to be 64, 10, 1e-5, and 0.01, respectively. For baselines, we consider 11 methods whose details can be found in Appendix B.1. Note that for comparison with those baselines using meta-features to measure the similarity between datasets, we extract meta-features for each dataset following the Table 1 in [23]. We compare in terms of the maximum classification accuracy achieved so far and the average rank over all datasets indicating the rank of a method (lower is better, please refer to [1] for more details).

Results We aim to improve the classification accuracy of Logistic Regression (LR) [12] on 100 selected OpenML [21] datasets. Please refer to more details of OpenML in Section C of Appendix.



(a) : Average rank (b) : wine+oh5.wc→ kr-vs-kp (c) : fri_c4_100+fri_c4_25 → mfeat-fourier

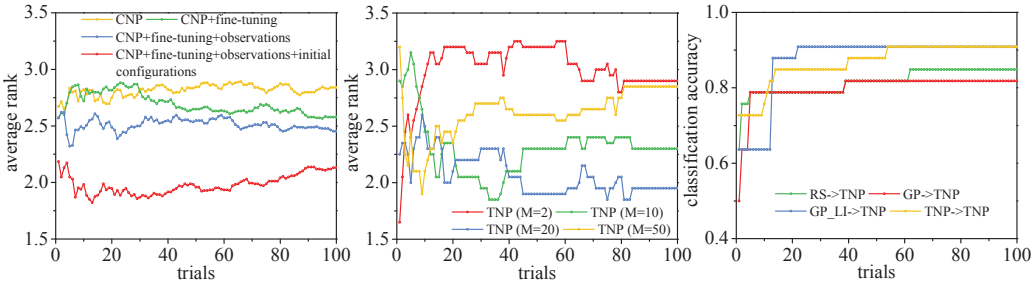
Figure 1: Average ranks over 100 datasets, and maximum accuracies achieved on two datasets.

The dimension of the hyperparameter space is four, including the learning rate $\eta \in [2^{-6}, 2^{0}]$ for SGD, the l2-regularization coefficient $r_2 \in [0, 1]$, the batch size $B \in [20, 2000]$, and the dropout ratio $\gamma \in [0, 0.75]$. Taking each dataset as the target, we randomly sample $M = 2$ of the 99 others as historical datasets. For each of the two historical datasets, we obtain its set of historical observations by running GPs on it to optimize the hyperparameters of LR within 100 trials. Figure 1a shows the average rank over all datasets – the proposed TNP consistently and significantly outperforms other baselines, especially over a wide range of OpenML datasets. We also randomly select two datasets and compare the maximum classification accuracies achieved so far by different algorithms in Figure 1b, 1c. Though the performance of all baselines varies from dataset to dataset, TNP quickly learns a remarkable configuration.

We also study the influence of different components on the performance of TNP. As shown in Figure 2a, first, fine-tuning the global kernel to be dataset-specific improves over directly applying CNP. Second, leveraging previous observations via the dataset-aware attention and learning a set of well-generalized initial configurations substantially boost the effectiveness of TNP. Figure 2b investigates the influence of the number of historical datasets, i.e., M – more datasets generally contribute more to improve the HPO. As mentioned above, the history set of each dataset is obtained by running GPs on it. Here we are motivated to study how the base method used to produce the history set, e.g., GPs here, influences the performance of TNP. Take the "lymph" dataset as an example. The results in Figure 2c further guard the effectiveness of TNP regardless of the base method. The TNP dependent on the history sets produced by a more effective base method, say TNP itself, is prone to outperform, provided with more insightful observations by the superior base method. Additional results on hyperparameter sensitivity, efficiency, the effectiveness of the learned similarity between datasets, and three computer vision datasets can be found in Section C and D of Appendix.

5 Conclusion

We introduced a novel end-to-end HPO method which leverages knowledge from past HPO observations on other datasets. With dataset-aware attention, TNP attentively borrows observations from those similar datasets. TNP, to the best of our knowledge, is the first to harness the collective power of transferring observations, parameters for the surrogate model, and well-generalized configurations for SMBO. In particular, TNP enjoys the advantages of NPs with high scalability.



(a) : Varying different components (b) : Varying the # of meta-datasets (c) : Varying the base method

Figure 2: Varying different components, the number of meta-datasets, and the base method in TNP.

References

[1] Rémi Bardenet, Mátyás Brendel, Balázs Kégl, and Michele Sebag. Collaborative hyperparameter tuning. In *ICML*, pages 199–207, 2013.

[2] James S Bergstra, Rémi Bardenet, Yoshua Bengio, and Balázs Kégl. Algorithms for hyper-parameter optimization. In *NIPS*, pages 2546–2554, 2011.

[3] Matthias Feurer, Benjamin Letham, and Eytan Bakshy. Scalable meta-learning for bayesian optimization. *arXiv preprint arXiv:1802.02219*, 2018.

[4] Matthias Feurer, Jost Tobias Springenberg, and Frank Hutter. Initializing bayesian hyperparameter optimization via meta-learning. In *AAAI*, 2015.

[5] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *ICML*, pages 1126–1135, 2017.

[6] Marta Garnelo, Dan Rosenbaum, Christopher Maddison, Tiago Ramalho, David Saxton, Murray Shanahan, Yee Whye Teh, Danilo Rezende, and SM Ali Eslami. Conditional neural processes. In *ICML*, pages 1690–1699, 2018.

[7] Marta Garnelo, Jonathan Schwarz, Dan Rosenbaum, Fabio Viola, Danilo J Rezende, SM Eslami, and Yee Whye Teh. Neural processes. *arXiv preprint arXiv:1807.01622*, 2018.

[8] Erin Grant, Chelsea Finn, Sergey Levine, Trevor Darrell, and Thomas Griffiths. Recasting gradient-based meta-learning as hierarchical bayes. In *ICLR*, 2018.

[9] Frank Hutter, Holger H Hoos, and Kevin Leyton-Brown. Sequential model-based optimization for general algorithm configuration. In *LION*, pages 507–523, 2011.

[10] Donald R Jones, Matthias Schonlau, and William J Welch. Efficient global optimization of expensive black-box functions. *Journal of Global Optimization*, 13(4):455–492, 1998.

[11] Hyunjik Kim, Andriy Mnih, Jonathan Schwarz, Marta Garnelo, Ali Eslami, Dan Rosenbaum, Oriol Vinyals, and Yee Whye Teh. Attentive neural processes. In *ICLR*, 2019.

[12] Yann LeCun, Léon Bottou, Yoshua Bengio, Patrick Haffner, et al. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[13] Marius Lindauer and Frank Hutter. Warmstarting of model-based algorithm configuration. In *AAAI*, 2018.

[14] Sinno Jialin Pan and Qiang Yang. A survey on transfer learning. *TKDE*, 22(10):1345–1359, 2009.

[15] Valerio Perrone, Rodolphe Jenatton, Matthias W Seeger, and Cedric Archambeau. Scalable hyperparameter transfer learning. In *NIPS*, pages 6846–6856, 2018.

[16] Nicolas Schilling, Martin Wistuba, Lucas Drumond, and Lars Schmidt-Thieme. Hyperparameter optimization with factorized multilayer perceptrons. In *ECML/PKDD*, pages 87–103, 2015.

[17] Jasper Snoek, Hugo Larochelle, and Ryan P Adams. Practical bayesian optimization of machine learning algorithms. In *NIPS*, pages 2951–2959, 2012.

[18] Jasper Snoek, Oren Rippel, Kevin Swersky, Ryan Kiros, Nadathur Satish, Narayanan Sundaram, Mostofa Patwary, Mr Prabhat, and Ryan Adams. Scalable bayesian optimization using deep neural networks. In *ICML*, pages 2171–2180, 2015.

[19] Jost Tobias Springenberg, Aaron Klein, Stefan Falkner, and Frank Hutter. Bayesian optimization with robust bayesian neural networks. In *NIPS*, pages 4134–4142, 2016.

[20] Kevin Swersky, Jasper Snoek, and Ryan P Adams. Multi-task bayesian optimization. In *NIPS*, pages 2004–2012, 2013.

[21] Joaquin Vanschoren, Jan N Van Rijn, Bernd Bischl, and Luis Torgo. Openml: networked science in machine learning. *ACM SIGKDD Explorations Newsletter*, 15(2):49–60, 2014.

[22] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *NIPS*, pages 5998–6008, 2017.

[23] Martin Wistuba, Nicolas Schilling, and Lars Schmidt-Thieme. Learning hyperparameter optimization initializations. In *DSAA*, pages 1–10, 2015.

[24] Martin Wistuba, Nicolas Schilling, and Lars Schmidt-Thieme. Hyperparameter optimization machines. In *DSAA*, pages 41–50, 2016.

[25] Martin Wistuba, Nicolas Schilling, and Lars Schmidt-Thieme. Two-stage transfer surrogate model for automatic hyperparameter optimization. In *ECML/PKDD*, pages 199–214, 2016.

[26] Dani Yogatama and Gideon Mann. Efficient transfer learning method for automatic hyperparameter tuning. In *AISTATS*, pages 1077–1085, 2014.