LEARNING NEURAL PROCESSES ON THE FLY

Anonymous authors
Paper under double-blind review

ABSTRACT

Deep neural networks (DNNs) have performed impressively on a wide range of tasks, but they usually require a significant number of training samples to achieve good performance. Thus, DNNs do not work well in low-data regimes because they tend to overfit a small dataset and make poor predictions. In contrast, shallow neural networks (SNNs) generally are robust against overfitting in low-data regimes and converge more quickly than DNNs, but they struggle to represent very complex systems. Hence, DNNs and SNNs have a complementary relationship, and combining their benefits can provide fast-learning capability with high asymptotic performance, as meta-learning does. However, aggregating heterogeneous methods with opposite properties is not trivial, as it can make the combined method inferior to each base method. In this paper, we propose a new algorithm called anytime neural processes that combines DNNs and SNNs and can work in both low-data and high-data regimes. To combine heterogeneous models effectively, we propose a novel aggregation method based on a generalized product-of-exports and a winner-take-all gate network. Moreover, we discuss the theoretical basis of the proposed method. Experiments on a public dataset show that the proposed method achieves comparable performance with other state-of-the-art methods.

1 INTRODUCTION

Deep neural networks (DNNs) have been increasingly used for various tasks. One of their notable properties is that the performance of DNNs improves exponentially as the amount of available data increases Krizhevsky et al. (2012); Hochreiter & Schmidhuber (1997); Goodfellow et al. (2014); Sutton & Barto (2018). However, they tend to overfit on small datasets and do not work well in low-data regimes Ravi & Larochelle (2017). Thus, they are called a data-driven method. In contrast, shallow neural networks (SNNs), such as linear models and kernel methods, tend to learn quickly and are robust against overfitting problems Rasmussen (2003); Chua et al. (2018); Kamthe & Deisenroth (2018). However, their asymptotic performance tends to converge to less optimal solutions and lags behind that of DNN-based methods Chua et al. (2018). Hence, SNNs and DNNs have a complementary relationship, and thus it is interesting and natural to synergize the two methods. A combined method is expected to learn quickly in low-data regimes (in the early stages) and achieve high asymptotic performance in high-data regimes (in the later stages of the learning process), as shown in Figure 1.

Meta-learning has a similar motivation to our method in that meta-trained models quickly learn a new task from only a few examples, and they continue to adapt as more data become available Schmidhuber (1987); Thrun & Pratt (2012). It essentially leverages information gathered from prior related tasks to learn more effectively in a novel task, and it demonstrates good performance in various fields such as few-shot classification and RL problems Finn et al. (2017); Santoro et al. (2016); Nichol et al. (2018). However, meta-training needs a significant amount of tasks that are similar to the target task. In real-world scenarios, it is not trivial to find and define training tasks similar to the target tasks. In addition, each task should be mutually exclusive such that no single model can solve all the tasks at once, which requires users to pay attention to designing the tasks, such as removing task identifying information or shuffling the index for labels Yin et al. (2019). These constraints make meta-learning inapplicable to some domains where the boundaries of tasks cannot be determined beforehand Xu et al. (2020).

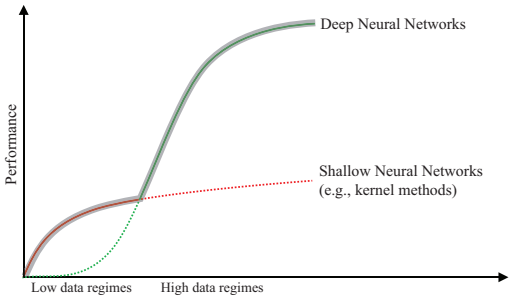


Figure 1: Distinctive properties of the proposed method. The performance of deep neural networks (DNNs) tends to improve exponentially as the amount of data increases. However, when the amount of data is small, DNNs fail to achieve generalized performance during the test phase because they are typically overfitted to the training dataset. In contrast, shallow neural networks converge quickly and are robust to overfitting. However, their performance reaches a non-optimal saturation point early, no matter how much more data the algorithms use. The performance of the proposed method follows the shaded line without significant performance degradation.

We approach meta-learning from a different perspective and propose an ensemble-based method that learns with a few training examples and directly achieves reasonable performance on the test dataset. As more training data become available, the performance of the proposed method improves significantly, as with DNN-based methods. To achieve this goal, we must solve two challenges. The first is to prevent overfitting on a small amount of data in a low-data regime before collecting enough data for the DNNs. We use Bayesian methods because they offer robustness to overfitting. However, Bayesian methods generally have challenging inferences and additional computational costs in a high-data regime Hensman et al. (2013); Rasmussen & Williams (2006). Therefore, the computational burden can be significantly reduced by transitioning from Bayesian methods to DNNs sometime during learning. When to transition is not a trivial question. The second challenge is to combine heterogeneous models without a significant performance drop. Ensemble learning is a representative method for combining learning algorithms to improve performance Dietterich (2000); Huang et al. (2017); Wasay et al. (2020). The core part is how to assign a proper weight to each expert model. Typically, predictions from independently trained models are averaged with variation induced by bagging or randomization. However, simple averaging does not apply to our problem because insufficiently trained DNN models degrade the performance of the combined model in low-data regimes. Hence, it is important to assign a proper weight to each base model according to its performance.

To handle these problems, we propose anytime neural processes (ATNPs) that perform fast learning in the earlier stages and accomplish high performance in the later stages of the learning algorithms. This capability is essential for robots to learn primitive skills such as collision avoidance and navigation, especially in *unknown environment* Kahn et al. (2017); Seker et al. (2019). For low-data regimes, we exploit Gaussian processes (GPs) as Bayesian methods to reduce overfitting on a small dataset. GPs efficiently learn tasks from small datasets and generate a plausible solution without requiring a large number of iterations because the standard GPs are mathematically equivalent to infinitely wide SNNs Neal (1994). Therefore, the costly meta-training process can be replaced by GPs heuristically Hinton & Salakhutdinov (2008). For high-data regimes, conditional neural processes (CNPs) are used to achieve high asymptotic performance. CNPs are a DNN-based method that learns an approximation to a stochastic process Garnelo et al. (2018a;b). To combine the advantages of both stochastic models, we introduce a novel aggregating network based on a probabilistic ensemble method Dietterich (2000) and a winner-take-all (WTA) gate network Cios & Shields (1997); Eigen et al. (2014). The proposed aggregating method enables ATNPs to converge to the CNPs as additional learning iterations are performed. Thus, ATNPs can take full advantage of GPs in low-data regimes and guarantee that the switch from GPs to CNPs occurs without significant performance degradation as soon as the performance of CNPs exceeds that of the GPs.

The contributions of this paper can be summarized as follows: 1) We propose an ATNP that quickly identifies a feasible solution, then improves its optimality over all data regimes. The proposed method combines GPs and CNPs in a novel ensemble method. 2) For effective ensemble learning, we introduce a novel gate network that guarantees that the transition from GPs to CNPs occurs without significant performance degradation. The theoretical discussion of the transition is provided. 3) We experimentally demonstrate that the proposed method outperforms CNPs in low-data regimes and is comparable with CNPs in high-data regimes on a public dataset. In addition, the proposed method achieves comparable performance to meta-learning algorithms.

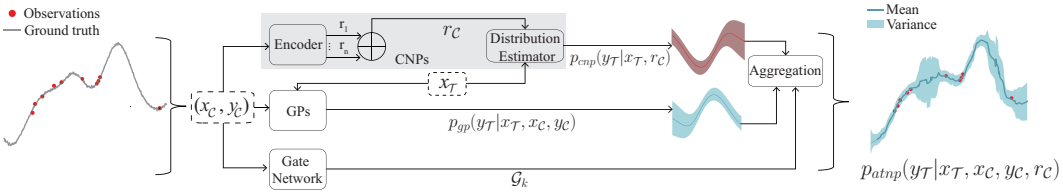


Figure 2: The proposed anytime neural processes (ATNP) framework includes neural (CNPs) and Gaussian (GPs) processes with gate networks. Inputs are shared among the components and each CNPs and GPs estimates factorized predictive distributions. The gate networks controls the importance $\mathcal{G}_k$ for each estimate and generates aggregated predictive distributions.

2 PRELIMINARIES

2.1 GAUSSIAN PROCESSES FOR REGRESSION

A GP regressor predicts a target output $y_* \in \mathbb{R}^{d_y}$ of a target input $x_* \in \mathbb{R}^{d_x}$ given a set of observations of size $(x_C, y_C) = (x_i, y_i)_{i \in C}$, where $y_* = f(x_*) + w$ with $w \sim \mathcal{N}(0, \sigma_w^2)$ and $f(\cdot)$ follows a GP Rasmussen & Williams (2006). Specifically, given the target input x_* and a set of observations of size n, the conditional distribution of the target y_* becomes

$$
\begin{aligned}
&p_{gp}(y_*|x_*, x_C, y_C) \sim \mathcal{N}(\hat{\mu}(x_*), \hat{\sigma}^2(x_*)), \\
&\hat{\mu}(x_*) = k(x_*, x_C)(k(x_C, x_C) + \sigma_w^2 I)^{-1} y_C, \\
&\hat{\sigma}^2(x_*) = k(x_*, x_*) - k(x_*, x_C)(k(x_C, x_C) + \sigma_w^2 I)^{-1} k(x_C, x_*),
\end{aligned} \tag{1}
$$

where $k(x_*, x_C)$ is a covariance vector between the target input and the observations inputs, $k(x_t, x_t) \in \mathbb{R}$, and $k(x_C, x_C) \in \mathbb{R}^{n \times n}$ is a kernel matrix. Standard GPs have computational complexity, $\mathcal{O}(n^3)$ and performance remains restricted by prior kernel type, often called kernel selection problem.

2.2 CONDITIONAL NEURAL PROCESSES

CNPs are a DNN-based method that learns approximations to stochastic processes Garnelo et al. (2018a;b). There is no need to find appropriate priors because CNPs learn an implicit kernel directly from the data, solving the kernel selection problem. Hence, CNPs can learn complex learning tasks efficiently in high-data regimes. Additionally, CNPs achieve flexible and fast prediction for stochastic processes because once trained, CNP inference complexity is linear in observation size, with a runtime complexity of $\mathcal{O}(n + m)$ to make m predictions with n observations Garnelo et al. (2018a;b). Specifically, CNPs learn the approximation of conditional distributions

$$
p_{cnp}(y_T|x_T, x_C, y_C) = p_{cnp}(y_T|x_T, r_C), \quad r_C = r_\zeta(x_C, y_C), \tag{2}
$$

where r_ζ and $(x_T, y_T) = (x_i, y_i)_{i \in T}$ are a neural network and a set of targets, respectively. The likelihood $p_{cnp}(y_T|x_T, r_C)$ is modeled by a Gaussian distribution factorized with means and variances determined by passing x_T and r_C. This structure is based on the variational auto-encoder Kingma & Welling (2014); Sohn et al. (2015); hence the likelihood corresponds to a decoder, and r_ζ can be considered as an encoder.

We focus on the standard regression problem. Conditioned on observation (x_C, y_C), GPs can be defined as joint Gaussian distributions $p_{gp}(y_T|x_T, x_C, y_C)$ of target y_T. CNPs can be considered as a DNN-based method that learns approximations to GPs Garnelo et al. (2018a;b). Both p_{gp} and p_{cnp} distributions can be combined analytically because they are compatible. With the aforementioned distributions, the goal is to learn the joint distribution $p_{atnp}(y_T|x_T, x_C, y_C, r_C)$ that performs well in an all-data regime.

3 PROPOSED ANYTIME NEURAL PROCESSES

This section describes the proposed ATNPs that combine GPs and CNPs. ATNPs take *observations* as inputs and estimate predictive *target* distributions. The proposed framework achieves optimality

when an aggregated model converge to CNPs in the training phase, which guarantees running time complexity $\mathcal{O}(n+m)$ for CNPs. We modify the generalized product-of-experts (gPoE) by replacing the mixing coefficient with a gate network with WTA. Figure 2 shows the proposed framework.

3.1 ENSEMBLE

Ensemble methods combine multiple model predictions to improve accuracy and reduce variance. However, we use ensemble methods to combine GPs and CNPs in a way that each expert plays a different role at different time. Probabilistic models commonly employ mixture of experts (MoEs) and product-of-experts (PoEs) frameworks Jacobs et al. (1991); Hinton (2002). MoEs perform voting among experts and provide good performance when the experts have high variance, whereas PoEs allow experts to specialize in analyzing one particular aspect of the problem. However, PoEs tend to be overconfident Ng & Deisenroth (2014). This means that the prediction variance of PoEs shrinks to 0 when the number of training data goes to infinity. Thus, PoE models are not appropriate for safety-critical systems such as self-driving cars. To sidestep this problem, we exploit the gPoEs to estimate predictive distributions Cao & Fleet (2014).

3.1.1 GENERALIZED PRODUCT OF EXPERTS

PoEs models assume the experts to be independent. Given a training data set $\mathcal{D} = \{\mathbf{x}, \mathbf{y}\}$ with inputs $\mathbf{x} = \{x_i\}_{i=1}^N$ and observations $\mathbf{y} = \{y_i\}_{i=1}^N$, the marginal likelihood $p(\mathbf{y}|\mathbf{x}, \theta)$ is approximated by $p(\mathbf{y}|\mathbf{x}, \theta) \approx \prod_{k=1}^K p_k(\mathbf{y}^{(k)}|\mathbf{x}^{(k)}, \theta)$, where p_k is the predictive distribution of the kth expert, and K is the total number of experts, and hence we can partition $\mathcal{D}$ into K sets $\mathcal{D} = \cup_{k=1}^K D^{(k)}$. For the regression problem $y = f(x) + \epsilon$, gPoEs return a distribution for an unknown function $f_* = f(x_*)$ as $p_{gpoe}(f_*|x_*, \mathcal{D}) = \prod_{k=1}^K p_k^{\beta_k}(f_*|x_*, \mathcal{D}^{(k)})$, where the β_k weights expert contributions. For GP experts, p_k has normal distribution with mean $\mu_k(x_*)$ and variance $\sigma_k^2(x_*)$, $k = 1, ..., K$. Therefore, the joint prediction $p_{gpoe}(f_*|x_*, \mathcal{D})$ has mean μ_*^{gpoe} and variance σ_*^{gpoe} satisfying $\mu_*^{gpoe} = (\sigma_*^{gpoe})^2 \sum_{k=1}^K \beta_k \sigma_k^{-2}(x_*)\mu_k(x_*)$ and $(\sigma_*^{gpoe})^{-2} = \sum_{k=1}^K \beta_k \sigma_k^{-2}(x_*)$.

Cao and Fleet Cao & Fleet (2014) suggested to set β_k to the differential entropy between the prior and the posterior. However, this method would make our proposed algorithm have a $\mathcal{O}((n+m)^3)$ computation cost throughout training process and in test phase. Another option could be to set β_k constant e.g. $\beta_k = 1/K$, which would ensure stable performance Deisenroth & Ng (2015), but that still does not solve the computational cost problem. Therefore, we propose an aggregation method based on gPoE that fulfills three requirements: (i) β_k is dependent of input data or CNPs byproducts. In particular, β_k is independent from GPs outputs, saving GPs computational costs after the gate network reaches the optimality. (ii) β_k corresponding to CNPs converges to 1 to achieve optimality. (iii) β_k is determined by each expert's performance. We replace β_k with the gate network $\mathcal{G}$ and the combined model becomes

$$p(f_*|x_*, \mathcal{D}) = \prod_{k \in \mathcal{K}} p_k^{\mathcal{G}_k}(f_*|x_*, \mathcal{D}), \tag{3}$$

where $\mathcal{K} = \{GPs, CNPs\}$. Hence, we exploit GPs and CNPs as experts and train them jointly via gradient descent in an end-to-end fashion. The predictive mean and variance are $\mu_*^\Upsilon = (\sigma_*^\Upsilon)^2 \sum_{k \in \mathcal{K}} \mathcal{G}_k \sigma_k^{-2}(x_*)\mu_k(x_*)$ and $(\sigma_*^\Upsilon)^{-2} = \sum_{k \in \mathcal{K}} \mathcal{G}_k \sigma_k^{-2}(x_*)$, respectively. The final predictive distribution is

$$\mathcal{Q}_\theta(x_*) = p_{atnp}(y_*|x_*, \mu_*^\Upsilon, \sigma_*^\Upsilon, \theta), \tag{4}$$

where θ refers to learnable ATNPs parameters, including CNP parameters θ_{cnp}, gate network parameters θ_{gate}, and GPs expert hyperparameters θ_{gp} in Table 4 in the appendix. We only employ Eq. (4) to train the whole network, individual experts are implicitly trained by minimizing the loss function (see Section 3.3).

3.2 GATE NETWORK

Gate networks control information flows in neural networks Hochreiter & Schmidhuber (1997). Jacobs *et al.* Jacobs et al. (1991) employed gate networks to weight experts for ensemble methods. A trained gate network appropriately mixes expert network outputs to produce a final output. We use a gate network to weight GPs and CNPs in the proposed framework, leveraging WTA where

some expert always has the largest weight Cios & Shields (1997); Eigen et al. (2014). Although WTA is generally undesirable for ensemble methods Eigen et al. (2014); Peralta et al. (2019), it guarantees optimality for our problem by letting $\mathcal{G}_{gp}$ converge to 0 in training. Softmax function can be considered as a softened version of WTA procedure, hence we employ a gate network that incorporate the softmax function $\mathcal{G}_k(\mathbf{v}) = \frac{e^{\alpha v_k}}{\sum_{j=1}^{K} e^{\alpha v_j}}$, where $k = 1, ..., K$, $\mathbf{v} = (v_1, ..., v_K) \in \mathbb{R}^K$ and $\alpha \geq 0$. When $\alpha \to \infty$, the softmax function becomes the classic WTA where output from the largest input 1 and other outputs are 0. At the other extreme, all outputs tend to $\frac{1}{K}$ as $\alpha \to 0$. Thus, we define softmax with an embedding layer Ψ such that $\mathcal{G}_k(\mathbf{v}) = \frac{e^{\alpha \Psi_k(v)}}{\sum_{j=1}^{K} e^{\alpha \Psi_j(v)}}$. To render $\mathcal{G}_{gp}$ close to 1 at the beginning of training, we apply bias ($\eta_{gp} \gg \eta_{np}$) to neural network outputs rather than controlling the hyperparameter α to avoid numerical instability, and the gate network $\mathcal{G}_k$ takes *observations* (x_C, y_C),

$$\mathcal{G}_k(x_C, y_C) = \frac{e^{\Psi_k(x_C, y_C) + \eta_k}}{\sum\limits_{j \in \mathcal{K}} e^{\Psi_j(x_C, y_C) + \eta_j}}, \tag{5}$$

where $\Psi_k(x_C, y_C)$ refers to the embedding vector for each expert. Thus, $\mathcal{G}_{gp}$ converges to 0 around the time when the likelihood of NPs exceeds that GPs because the convergence is self-reinforcing. We discuss the theoretical basis for convergence in Section 3.4.

3.3 Training

The predictive distribution model Eq. (4) is trained in an end-to-end manner to jointly optimize experts and gate network, i.e., CNPs parameters and GPs hyperparameters are simultaneously updated at each iteration. Let $\mathcal{P}$ be a probability distribution over a class of function $\mathcal{F}$, where $\mathcal{F} = \{f | f : X \to Y\}$ is a stochastic process. We sample $f \sim \mathcal{P}$ to generate training dataset $O = \{(x_i, y_i)\}_{i=1}^{N}$, define subset $O_C = \{(x_i, y_i)\}_{i=1}^{n}$, $n \sim$ uniform$[1, ..., N]$ as *observations*, predict O conditioned on O_C and minimize the conditional negative log likelihood (NLL)

$$\mathcal{L}(\theta) = -\mathbb{E}_{f \sim P}\Big[\mathbb{E}_N\Big[\log Q_\theta(\{y_i\}_{i=1}^{N} | O_C, \{x_i\}_{i=1}^{N})\Big]\Big] \tag{6}$$

using the Adam optimizer Kingma & Ba (2015).

3.4 Discussion

To understand gate network convergence, we analyzed the transition from GPs to CNPs expert theoretically. The transition occurs when one expert represents the data distribution better than the other in our framework. Models with high degrees of freedom or capacity usually express a diverse set of hypotheses, hence a more complex model has more flexibility to find target functions that fit the data well. In statistical learning theory, various complexity measures for machine learning have been proposed, including Vapnik–Chervonenkis (VC) dimension Vapnik (2013) and Rademacher complexity Bartlett & Mendelson (2002). However, these classical complexity measures are generally unsuitable for high expressive model such as DNNs Zhang et al. (2017). This is also true for the proposed framework in terms of VC dimension with following a definition.

Definition 1 *For any sample size N, the Vapnik-Chervonenkis (VC) dimension of a hypothesis set $\mathcal{H}$, denoted by $d_{vc}(\mathcal{H})$ or simply d_{vc}, is the largest value of N for which $m_{\mathcal{H}}(N) = 2^N$. If $m_{\mathcal{H}}(N) = 2^N$ for all N, then $d_{vc}(\mathcal{H}) = \infty$.*

This definition is universal in the sense that it applies to all hypothesis sets, learning algorithms, probability distribution, and binary target functions Abu-Mostafa et al. (2012). Specifically, we can obtain a finite VC dimension for the CNPs expert Abu-Mostafa et al. (2012). In contrast, GPs have infinite VC dimension because non-parameteric models can be viewed as infinitely complex ML models Evgeniou & Pontil (1999); Reeb et al. (2018). The transition from CNPs to GPs expert should occur according to VC dimension, but this contradicts our results. Therefore, the convergence cannot be explained with VC dimension. Instead, we explain the convergence in two respects as follows.

Expressivity gap between the CNPs and the GPs: Neural network expressivity in ML can be defined as how the network architecture, e.g. width, depth, connectivity, affects the resulting function

properties Poole et al. (2016); Raghu et al. (2017). One expressivity principle is that the number of hidden layers has larger impact on expressive power than the number of neurons in a hidden layer for neural networks with bounded size. Hence, DNNs learn complex feature hierarchies over input space in ways that shallow networks (one hidden layer and the same number of neurons) cannot Delalleau & Bengio (2011); Montufar et al. (2014). Therefore, shallow networks are usually not good at generalization even though they can approximate any function Cybenko (1989). The standard GP is mathematically equivalent to an infinitely wide shallow network Neal (1994), hence its expressive power is lower than generic DNNs. For example, standard GPs with commonly used kernels cannot describe well data incorporating non-smooth functions, e.g. a step function Calandra et al. (2016); Pang et al. (2019).

Different training regimes of CNPs and GPs: CNPs are trained on many different functions, whereas GPs are usually trained on observations from a single function. In our training setting, the GPs learn a new task (or a new data distribution) at each iteration from their point of view. It is likely to forget the knowledge (i.e., learned hyperparameters) GPs obtained from the previous iteration. Thus, the GPs do not fit various functions well even if the training progresses. This is similar to the catastrophic forgetting problem where a model's performance on learned tasks abruptly degrades when trained for a new task in continual learning McCloskey & Cohen (1989).

4 Related Works

A GP is a popular Bayesian method which is used for learning tasks with few data samples because it provides a mathematical framework to incorporate prior knowledge about tasks Rasmussen & Williams (2006); McNeish (2016). GPs have often been used in low-data regimes because of their fast-learning capability and robustness to overfitting. In Deisenroth & Rasmussen (2011), a GP-based policy search method (PILCO) was proposed to solve data-inefficiency issues, which are a long-standing problem in trial-and-error learning processes. PILCO can learn from scratch in only a few trials and does not rely on expert knowledge such as task-specific prior knowledge or demonstrations. In Kamthe & Deisenroth (2018), model prediction control (MPC) with GPs was proposed to reduce the number of interactions with the environment in RL. However, it is still difficult for GPs to learn complex patterns with large datasets, and their asymptotic performance is usually lower than that of DNN-based methods because their expressive power is lower than generic DNNs Calandra et al. (2016); Pang et al. (2019).

The proposed method has a similar motivation to that of meta-learning. Meta-learning has been proposed to quickly learn new tasks using few samples by first learning prior knowledge from similar tasks Lake et al. (2015). However, meta-learning needs to define and obtain training tasks similar to a target task for meta-training, which is not trivial in the real world. In contrast, the proposed method does not need to find training tasks similar to the target tasks. The learning strategy of the proposed method is similar to Sahoo et al. (2018) in such a way that our method leverages GPs for rapid performance improvement in the early stages of learning, then gradually shifts to DNNs automatically when more data have been gathered. In Sahoo et al. (2018), a hedge algorithm Freund & Schapire (1997) was used to combine predictions with multiple outputs from shallow to deep layers.

5 Experiments

The proposed method was trained on multiple realizations of the stochastic process as in Garnelo et al. (2018a;b); Kim et al. (2019). We evaluate the proposed method on a regression task using CNPs and attentive NPs (ANPs) as baselines. The algorithm was implemented using tensorflow Abadi et al. (2016). See appendix A.1 for architecture details.

5.1 1-D function regression task

We followed the experiment setup for a regression task used in Garnelo et al. (2018a;b); Kim et al. (2019) and evaluated the proposed method on the classical one-dimensional regression used as a common baseline for GPs. The generated datasets comprised functions generated from GPs with exponentiated quadratic kernel and small likelihood noise. Each iteration randomly selected the

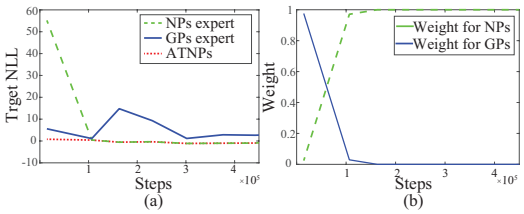


Figure 3: Quantitative ATNPs framework results with respect to training iterations. (a) Target NLL given observations. (b) Mixing weights for each expert estimated by the gate network.

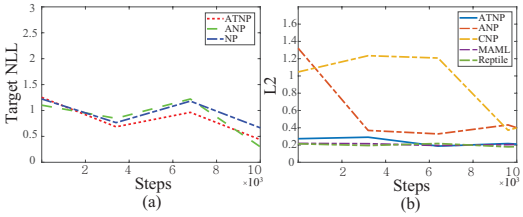


Figure 4: Comparison of the ATNPs with the baseline models. (a) Target NLL and (b) L2 for the mean estimates.

number of $observations$ and $targets$, and each x-value was sampled uniformly from $[-2, 2]$. The ATNPs outperformed the CNPs in the early training because the CNPs learned to convert their millions of network parameters into a predictive distribution from scratch, whereas the GP expert of the ATNPs used similarity measures from the kernel function to predict a distribution for each test point. As training advanced, the CNP expert of the ATNPs had a larger impact on the predictive distribution. ATNPs have the same computational efficiency as the CNPs expert after convergence because the GPs expert can be removed.

Figure 3(a) summarizes quantitative results. We evaluated target NLL given observations $\frac{1}{|T|}\sum_{i\in T}\mathbb{E}[\log p(y_i|x_i, r^*(x_C, y_C, x_i))]$ for each expert and the aggregated model. The GPs expert achieved lower NLL than the CNPs expert and the gate function assigned greater weight to GPs expert in early stage training. Therefore, the aggregated model was able to estimate a plausible predictive distribution even at a single gradient step. As training progresses, the CNPs expert improved its performance, whereas the GPs expert achieved similar or worse performance than the initial result because it had fixed kernel functions. Figure 3(b) shows that the gate network converged to the optimal as soon as the CNPs expert outperformed the GPs expert. Figure 4 compares the proposed ATNPs framework and CNPs and ANPs models Garnelo et al. (2018a;b); Kim et al. (2019). To ensure fair comparison, the CNPs expert had the same architecture as for the baseline, including the same number of parameters. Figure 4(a) shows that the proposed ATNPs framework achieved similar performance to ANPs model because the variance estimates affected NLL more than mean estimates do in the early stage, hence large variance estimates compensated for poor mean estimates. Therefore, we adopted L2 as an additional performance metric, calculating the distance between mean estimates and ground truth. Figure 4(b) shows that the ATNPs framework had much lower L2 than the baseline models. Both performance curves based on NLL and L2 did not satisfy a monotonicity property because of the training regime using stochastic gradient descent optimizer. However, the ATNPs had a monotonicity property in the sense that the transition from GPs expert to NPs expert was guaranteed.

Table 1: Processing time per iteration in training phase on 1D synthetic data.

Models	Mean (ms)	Variance (ms)
ATNPs	9.5184	0.1123
ANPs	8.7044	0.1186
CNPs	**6.0**	**0.0159**

We also evaluated processing time for the proposed method and a family of CNPs. Table 1 presents the means and variances of processing time per iteration before the gate network converged in training phase. The proposed method was the slowest among them because of the added computation costs from the GPs expert as well as the CNP expert. However, the baseline models required tens of thousand iterations to produce results comparable to ATNPs results obtained in few iterations as shown in Figure 4(b) and the computational complexity was the same as the baselines after the convergence. Therefore, the ATNPs can find a feasible solution within a few seconds with respect to absolute time.

Comparison with meta-learning : To show the *fast-learning* property of our method, we compared our method to meta-learning algorithms on the 1-D function regression problem. We used MAML Finn et al. (2017) and Reptile Nichol et al. (2018), which are representative gradient-based meta-learning methods. To conduct a fair comparison, we trained these methods with the same dataset used in Section 5.1. Following the meta-training setup used in Finn et al. (2017); Nichol et al. (2018), we sampled a task $\mathcal{T}_i$ from task distribution $p(\mathcal{T})$. Each task included a training set $\mathcal{D}_{(\mathcal{T}_i,train)}$ and a test set $\mathcal{D}_{(\mathcal{T}_i,test)}$. In the case of k-shot learning, the training set $\mathcal{D}_{(\mathcal{T}_i,train)}$ had k training

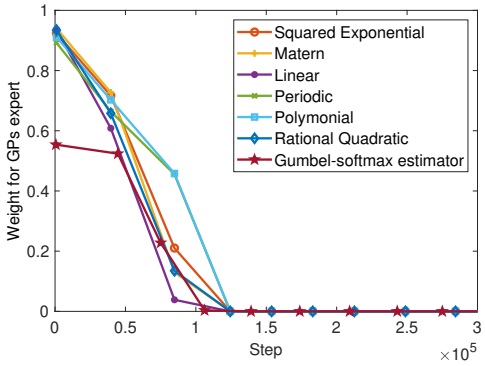


Figure 5: Quantitative results with respect to iteration for the proposed gate network with various kernel functions and Gumbel-softmax estimator.

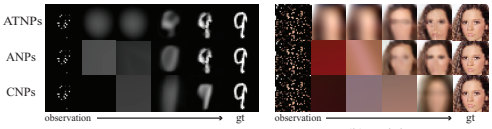
Table 2: ATNPs performance with different expert combinations.

Models	Mean	Variance
GPs + ANPs	**0.2918**	**0.0163**
GPs + CNP	0.3419	0.0376
SGPs + ANPs	0.3048	0.0425
SGPs + CNPs	0.3762	0.0709
ANPs	0.4502	0.0513
CNPs	0.6812	0.0748

samples. During the training phase, each task-specific learner updated its parameters by gradient descent using the loss evaluated with the training data $\mathcal{D}_{(\mathcal{T}_i,train)}$ in the inner loop. Then, the meta-learning across tasks was performed using the loss evaluated with $\mathcal{D}_{(\mathcal{T}_i,test)}$ in the outer loop. In our problem, a task corresponded to a function randomly generated from the GPs used in Section 5.1. The randomly selected $observations$ and $targets$ corresponded to $\mathcal{D}_{\mathcal{T}_i,train}$ and $\mathcal{D}_{\mathcal{T}_i,test}$, respectively. Specifically, we used 10,000 tasks whose training set and test set included 10 and 100 samples, respectively. We trained both MAML and Reptile using an Adam optimizer Kingma & Ba (2015) and set an inner loop and an outer loop learning rate to 0.01 and 0.001, respectively. We used only the L2 distance as a metric because both meta-learning algorithms are not stochastic methods. Figure 4(b) shows that the proposed method achieved comparable results in the early stage of training, which means that the GP expert of the ATNPs can replace the costly meta-learning process effectively. As training advances, the ATNPs achieved similar performance to the meta-learning baselines even though the proposed method estimated not only target means but also their variances.

Gate network convergence : The proposed gate network ensures that ATNPs converge asymptotically to the CNPs expert, as shown in Figure 3(b) where $\mathcal{G}_{gp}$ evolves from 1 to 0 and $\mathcal{G}_{cnp}$ vice versa. Once the gate network converges to the optimal state, the reverse transition is disallowed by WTA. This is self-reinforcing because a better expert is trained more rapidly and thus is more heavily weighted by the gate network. The CNPs expert performance in training always exceeds GPs expert at some point, as shown in Figure 3(a), and discussed in Section 3.4. Figure 5 compares the proposed gate network with baseline to demonstrate its effectiveness. We defined baselines by replacing the proposed gate network with a binary value gate network based on the Gumbel-Softmax estimator Jang et al. (2017); Maddison et al. (2017), originally developed to generate approximated and differentiable samples for categorical latent variables in a stochastic computational graph. It is often used for gate networks to push their outputs to the boundaries of their ranges Li et al. (2018). Given a probability distribution over k categories with trainable parameters $\kappa_1, \kappa_2, ..., \kappa_K$, the Gumber-Softmax estimator generates an approximate one-hot sample y with $y_i = \frac{\exp((\log \kappa_i + q_i)/\tau)}{\sum_{j=1}^{k} \exp((\log \kappa_i + q_i)/\tau)}$, for $i = 1, ..., k$, where τ is the temperature and q_i is independently sampled from the Gumbel distribution, $q_i = -\log(-\log U_i), U_i \sim$ Uniform (0,1). For baseline convergence, we trained the Gumbel-softmax estimator to push output values towards 0 or 1. However, $\mathcal{G}_{gp} \approx 0.5$ in early training as shown in Figure 5, hence the untrained CNPs expert affects the final predictive distribution, which renders poor results initially compared to using GPs alone. In our problem, GPs expert kernel function type does not affect the final trained model, and the CNPs expert is optimal and eventually achieved. To show that ATNPs converge to the CNPs expert regardless of GPs kernel type, we trained the ATNPs with commonly used kernel functions Rasmussen & Williams (2006) in Table 4 in the appendix. Hyperparameters for all kernel types were trainable except the exponent parameter for the polynomial kernel. Figure 5 shows that all the models converge to the NPs expert, hence optimality was achieved.

Model-agnostic : The ATNPs is general and model-agnostic in the sense that it is compatible with any probabilistic model with mean and covariance, and it does not place constraints on the DNN architecture. Therefore, the proposed method can leverage various GPs and advanced CNPs variants



(a) MNIST (b) CelebA

Figure 6: Qualitative results on both MNIST and CelebA datasets. For each image, first and last columns refer to context inputs and ground truths respectively. Each row shows how the prediction on full target pixels evolves against training iterations.

Table 3: Average of L2 distance between estimated means and ground truths for all of the pixels on test images before the gate network is converged.

Dataset	ATNP	ANP	CNP
MNIST	**0.1849**	0.1955	0.2171
CelebA	**0.1320**	0.1844	0.2143

for better performance. We demonstrate the proposed method equipped with various GPs expert including SGPs Titsias (2009); Hensman et al. (2013) and CNPs expert. Table 2 shows L2 mean and variance before gate network converge to optimal. All the proposed methods with various combinations outperform a family of CNPs because of the GPs expert. Among the combinations, GPs and ANPs combination achieves the best performance because the SGPs approximate standard GPs for efficient computation. In addition, as pointed out in Attentive NPs Kim et al. (2019), ANPs have performance improved compared with CNPs by solving underfitting problem. Thus, performance improvement for each expert leads to overall ATNPs performance improvement.

5.2 2-D FUNCTION REGRESSION TASK

We tested the proposed method on a large-scale image dataset. The ATNPs map a 2D pixel location to its pixel intensity as a regression problem. The input x_i was a pixel coordinates normalized to $[0, 1]^2$, and the output was a normalized pixel intensity $y_i \in \mathbb{R}^1$ for grayscale and $y_i \in \mathbb{R}^3$ for RGB. We exploited ANPs as an expert and trained the ATNPs on two different datasets (MNIST LeCun et al. (1998) and CelebA Liu et al. (2015)). The training procedure was the same for both datasets. At each step, we selected a subset of the pixels as observations. Conditioned on context pixels, the ATNPs was trained to predict the target pixels in the image. For MNIST dataset, we used the same model architectures used for 1D function regression. Figure 6(a) shows the result of three different models : CNPs, ANPs, and ATNPs. Each row refers to predictions of a test image against training iterations (from left to right).

The ATNPs gave high intensities to pixels around areas representing a number, whereas other models estimated uninformative intensities over all pixels in the early stage of training. As training iteration increased, the ATNPs incrementally improved their performances. The performance differences among models were visually more noticeable on CelebA dataset. The ATNPs predicted outlines of the faces with blurred color, whereas other models generated unrecognizable predictions even at the first iteration of training. For CelebA dataset, we modified GPs expert to predict multiple output variables (RGB channel). Even though there are some approaches for multi-output GPs considering correlation of multiple output variables Boyle & Frean (2005), in this paper, we simply modeled each output variables as independent from the others and treated the separately. We also compared ATNPs quantitatively to other models. The performance metric was pixel-wise mean squared errors for all of the pixels $\frac{1}{|\mathcal{D}_{test}|}\sum_{M\in\mathcal{D}_{test}}\frac{1}{|M|}\sum_{j\in M}(y_j-\hat{y}_j)$ over training steps. As show in Table 3, the ATNPs outperformed the others even though the ATNPs converges asymptotically to the ANPs.

6 CONCLUSION

This paper proposes an ATNP framework that quickly produces a plausible and suboptimal predictive distribution and then improves its optimality over all data regimes by ensembling GPs and CNPs. To combine heterogeneous models effectively, we propose a novel aggregation method based on a generalized product-of-exports and a WTA gate network, which guarantees that the transition from GPs to CNPs occurs. The experimental results showed that the proposed method could return predictions with accuracy comparable to GPs initially and converge to the optimal solution generated by CNPs as more computational resources and training data are allocated. Additionally, the proposed method achieved comparable performance to meta-learning algorithms without a costly meta-training process.

REFERENCES

Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. Tensorflow: A system for large-scale machine learning. In *Proceedings of the 12th USENIX conference on Operating Systems Design and Implementation*, pp. 265–283. USENIX Association, 2016.

Yaser S Abu-Mostafa, Malik Magdon-Ismail, and Hsuan-Tien Lin. *Learning from data*, volume 4. AMLBook New York, NY, USA:, 2012.

Peter L Bartlett and Shahar Mendelson. Rademacher and gaussian complexities: Risk bounds and structural results. *Journal of Machine Learning Research*, 3(3):463–482, 2002.

Phillip Boyle and Marcus Frean. Dependent gaussian processes. In *Advances in neural information processing systems*, pp. 217–224, 2005.

Roberto Calandra, Jan Peters, Carl Edward Rasmussen, and Marc Peter Deisenroth. Manifold gaussian processes for regression. In *2016 International Joint Conference on Neural Networks (IJCNN)*, pp. 3338–3345. IEEE, 2016.

Yanshuai Cao and David J Fleet. Generalized product of experts for automatic and principled fusion of gaussian process predictions. *Modern Nonparametrics 3: Automating the Learning Pipeline workshop at NIPS*, 2014.

Kurtland Chua, Roberto Calandra, Rowan McAllister, and Sergey Levine. Deep reinforcement learning in a handful of trials using probabilistic dynamics models. In *Advances in Neural Information Processing Systems*, pp. 4754–4765, 2018.

Krzysztof J Cios and Mark E Shields. The handbook of brain theory and neural networks: By micheal a. arbib (ed.), mit press, cambridge, ma, 1995, isbn 0-262-01148-4, 1118 pp, 1997.

George Cybenko. Approximation by superpositions of a sigmoidal function. *Mathematics of control, signals and systems*, 2(4):303–314, 1989.

Marc Deisenroth and Carl E Rasmussen. Pilco: A model-based and data-efficient approach to policy search. In *Proceedings of the 28th International Conference on machine learning (ICML-11)*, pp. 465–472, 2011.

Marc Peter Deisenroth and Jun Wei Ng. Distributed gaussian processes. In *Proceedings of the 32nd International Conference on International Conference on Machine Learning-Volume 37*, pp. 1481–1490. JMLR. org, 2015.

Olivier Delalleau and Yoshua Bengio. Shallow vs. deep sum-product networks. In *Advances in neural information processing systems*, pp. 666–674, 2011.

Thomas G Dietterich. Ensemble methods in machine learning. In *International workshop on multiple classifier systems*, pp. 1–15. Springer, 2000.

Joshua V Dillon, Ian Langmore, Dustin Tran, Eugene Brevdo, Srinivas Vasudevan, Dave Moore, Brian Patton, Alex Alemi, Matt Hoffman, and Rif A Saurous. Tensorflow distributions. *arXiv preprint arXiv:1711.10604*, 2017.

David Eigen, Marc'Aurelio Ranzato, and Ilya Sutskever. Learning factored representations in a deep mixture of experts. *ICLR workshop*, 2014.

Theodoros Evgeniou and Massimiliano Pontil. On the v γ dimension for regression in reproducing kernel hilbert spaces. In *International Conference on Algorithmic Learning Theory*, pp. 106–117. Springer, 1999.

Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pp. 1126–1135. JMLR. org, 2017.

Yoav Freund and Robert E Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of computer and system sciences*, 55(1):119–139, 1997.

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 *International Conference on Machine Learning*, pp. 1690–1699, 2018a.

Marta Garnelo, Jonathan Schwarz, Dan Rosenbaum, Fabio Viola, Danilo J Rezende, SM Eslami, and Yee Whye Teh. Neural processes. *ICML Workshop on Theoretical Foundations and Applications of Deep Generative Models*, 2018b.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pp. 2672–2680, 2014.

James Hensman, Nicolò Fusi, and Neil D Lawrence. Gaussian processes for big data. In *Proceedings of the Twenty-Ninth Conference on Uncertainty in Artificial Intelligence*, pp. 282–290, 2013.

Geoffrey E Hinton. Training products of experts by minimizing contrastive divergence. *Neural computation*, 14(8):1771–1800, 2002.

Geoffrey E Hinton and Russ R Salakhutdinov. Using deep belief nets to learn covariance kernels for gaussian processes. In *Advances in neural information processing systems*, pp. 1249–1256, 2008.

Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8): 1735–1780, 1997.

Gao Huang, Yixuan Li, Geoff Pleiss, Zhuang Liu, John E Hopcroft, and Kilian Q Weinberger. Snapshot ensembles: Train 1, get m for free. *arXiv preprint arXiv:1704.00109*, 2017.

Robert A Jacobs, Michael I Jordan, Steven J Nowlan, Geoffrey E Hinton, et al. Adaptive mixtures of local experts. *Neural computation*, 3(1):79–87, 1991.

Eric Jang, Shixiang Gu, and Ben Poole. Categorical reparametrization with gumble-softmax. In *International Conference on Learning Representations (ICLR 2017)*. OpenReview. net, 2017.

Gregory Kahn, Adam Villaflor, Vitchyr Pong, Pieter Abbeel, and Sergey Levine. Uncertainty-aware reinforcement learning for collision avoidance. *arXiv preprint arXiv:1702.01182*, 2017.

Sanket Kamthe and Marc Deisenroth. Data-efficient reinforcement learning with probabilistic model predictive control. In *International Conference on Artificial Intelligence and Statistics*, pp. 1701–1710. PMLR, 2018.

Hyunjik Kim, Andriy Mnih, Jonathan Schwarz, Marta Garnelo, S. M. Ali Eslami, Dan Rosenbaum, Oriol Vinyals, and Yee Whye Teh. Attentive neural processes. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019. URL https://openreview.net/forum?id=SkE6PjC9KX.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In Yoshua Bengio and Yann LeCun (eds.), *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015. URL http://arxiv.org/abs/1412.6980.

Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *stat*, 1050:1, 2014.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25:1097–1105, 2012.

Brenden M Lake, Ruslan Salakhutdinov, and Joshua B Tenenbaum. Human-level concept learning through probabilistic program induction. *Science*, 350(6266):1332–1338, 2015.

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

Zhuohan Li, Di He, Fei Tian, Wei Chen, Tao Qin, Liwei Wang, and Tie-Yan Liu. Towards binary-valued gates for robust lstm training. In *International Conference on Machine Learning*, pp. 3001–3010, 2018.

Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Deep learning face attributes in the wild. In *Proceedings of the IEEE international conference on computer vision*, pp. 3730–3738, 2015.

Chris J. Maddison, Andriy Mnih, and Yee Whye Teh. The concrete distribution: A continuous relaxation of discrete random variables. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017. URL https://openreview.net/forum?id=S1jE5L5gl.

Michael McCloskey and Neal J Cohen. Catastrophic interference in connectionist networks: The sequential learning problem. In *Psychology of learning and motivation*, volume 24, pp. 109–165. Elsevier, 1989.

Daniel McNeish. On using bayesian methods to address small sample problems. *Structural Equation Modeling: A Multidisciplinary Journal*, 23(5):750–773, 2016.

Guido F Montufar, Razvan Pascanu, Kyunghyun Cho, and Yoshua Bengio. On the number of linear regions of deep neural networks. In *Advances in neural information processing systems*, pp. 2924–2932, 2014.

Radford M Neal. Priors for infinite networks (tech. rep. no. crg-tr-94-1). *University of Toronto*, 1994.

Jun Wei Ng and Marc Peter Deisenroth. Hierarchical mixture-of-experts model for large-scale gaussian process regression. *stat*, 1050:9, 2014.

Alex Nichol, Joshua Achiam, and John Schulman. On first-order meta-learning algorithms. *arXiv preprint arXiv:1803.02999*, 2018.

Simon O'Callaghan, Fabio T Ramos, and Hugh Durrant-Whyte. Contextual occupancy maps using gaussian processes. In *2009 IEEE International Conference on Robotics and Automation*, pp. 1054–1060. IEEE, 2009.

Simon T O'Callaghan and Fabio T Ramos. Gaussian process occupancy maps. *The International Journal of Robotics Research*, 31(1):42–62, 2012.

Guofei Pang, Liu Yang, and George Em Karniadakis. Neural-net-induced gaussian process regression for function approximation and pde solution. *Journal of Computational Physics*, 384: 270–288, 2019.

Billy Peralta, Ariel Saavedra, Luis Caro, and Alvaro Soto. Mixture of experts with entropic regularization for data classification. *Entropy*, 21(2):190, 2019.

Ben Poole, Subhaneil Lahiri, Maithra Raghu, Jascha Sohl-Dickstein, and Surya Ganguli. Exponential expressivity in deep neural networks through transient chaos. In *Advances in neural information processing systems*, pp. 3360–3368, 2016.

Morgan Quigley, Ken Conley, Brian Gerkey, Josh Faust, Tully Foote, Jeremy Leibs, Rob Wheeler, and Andrew Y Ng. Ros: an open-source robot operating system. In *ICRA workshop on open source software*, volume 3, pp. 5. Kobe, Japan, 2009.

Maithra Raghu, Ben Poole, Jon Kleinberg, Surya Ganguli, and Jascha Sohl Dickstein. On the expressive power of deep neural networks. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pp. 2847–2854. JMLR. org, 2017.

Carl Edward Rasmussen. Gaussian processes in machine learning. In *Summer School on Machine Learning*, pp. 63–71. Springer, 2003.

Carl Edward Rasmussen and Christopher KI Williams. Gaussian processes for machine learning. *Gaussian Processes for Machine Learning, by CE Rasmussen and CKI Williams. ISBN-13 978-0-262-18253-9*, 2006.

S. Ravi and H. Larochelle. Optimization as a model for few-shot learning. In *ICLR*, 2017.

David Reeb, Andreas Doerr, Sebastian Gerwinn, and Barbara Rakitsch. Learning gaussian processes by minimizing pac-bayesian generalization bounds. In *Advances in Neural Information Processing Systems*, pp. 3337–3347, 2018.

Doyen Sahoo, Quang Pham, Jing Lu, and Steven CH Hoi. Online deep learning: learning deep neural networks on the fly. In *Proceedings of the 27th International Joint Conference on Artificial Intelligence*, pp. 2660–2666, 2018.

Adam Santoro, Sergey Bartunov, Matthew Botvinick, Daan Wierstra, and Timothy Lillicrap. Meta-learning with memory-augmented neural networks. In *International conference on machine learning*, pp. 1842–1850, 2016.

Jürgen Schmidhuber. *Evolutionary principles in self-referential learning, or on learning how to learn: the meta-meta-... hook*. PhD thesis, Technische Universität München, 1987.

Muhammet Yunus Seker, Mert Imre, Justus H Piater, and Emre Ugur. Conditional neural movement primitives. In *Robotics: Science and Systems*, 2019.

Kihyuk Sohn, Honglak Lee, and Xinchen Yan. Learning structured output representation using deep conditional generative models. In *Advances in neural information processing systems*, pp. 3483–3491, 2015.

Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

Sebastian Thrun and Lorien Pratt. *Learning to learn*. Springer Science & Business Media, 2012.

Michalis Titsias. Variational learning of inducing variables in sparse gaussian processes. In *Artificial Intelligence and Statistics*, pp. 567–574, 2009.

Vladimir Vapnik. *The nature of statistical learning theory*. Springer science & business media, 2013.

Abdul Wasay, Brian Hentschel, Yuze Liao, Sanyuan Chen, and Stratos Idreos. Mothernets: Rapid deep ensemble learning. In *Proceedings of the Conference on Machine Learning and Systems (MLSys)*, 2020.

Mengdi Xu, Wenhao Ding, Jiacheng Zhu, ZUXIN LIU, Baiming Chen, and Ding Zhao. Task-agnostic online reinforcement learning with an infinite mixture of gaussian processes. *Advances in Neural Information Processing Systems*, 33, 2020.

Mingzhang Yin, George Tucker, Mingyuan Zhou, Sergey Levine, and Chelsea Finn. Meta-learning without memorization. In *International Conference on Learning Representations*, 2019.

Yijun Yuan, Haofei Kuang, and Sören Schwertfeger. Fast gaussian process occupancy maps. In *2018 15th International Conference on Control, Automation, Robotics and Vision (ICARCV)*, pp. 1502–1507. IEEE, 2018.

Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. Understanding deep learning requires rethinking generalization. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*, 2017. URL https://openreview.net/forum?id=Sy8gdB9xx.