# AUTOSEM: Automatic Task Selection and Mixing in Multi-Task Learning

**Han Guo**      **Ramakanth Pasunuru**      **Mohit Bansal**

UNC Chapel Hill

{hanguo, ram, mbansal}@cs.unc.edu

## Abstract

Multi-task learning (MTL) has achieved success over a wide range of problems, where the goal is to improve the performance of a primary task using a set of relevant auxiliary tasks. However, when the usefulness of the auxiliary tasks w.r.t. the primary task is not known a priori, the success of MTL models depends on the correct choice of these auxiliary tasks and also a balanced mixing ratio of these tasks during alternate training. These two problems could be resolved via manual intuition or hyper-parameter tuning over all combinatorial task choices, but this introduces inductive bias or is not scalable when the number of candidate auxiliary tasks is very large. To address these issues, we present AUTOSEM, a two-stage MTL pipeline, where the first stage automatically selects the most useful auxiliary tasks via a Beta-Bernoulli multi-armed bandit with Thompson Sampling, and the second stage learns the training mixing ratio of these selected auxiliary tasks via a Gaussian Process based Bayesian optimization framework. We conduct several MTL experiments on the GLUE language understanding tasks, and show that our AUTOSEM framework can successfully find relevant auxiliary tasks and automatically learn their mixing ratio, achieving significant performance boosts on several primary tasks. Finally, we present ablations for each stage of AUTOSEM and analyze the learned auxiliary task choices.

## 1 Introduction

Multi-task Learning (MTL) (Caruana, 1997) is an inductive transfer mechanism which leverages information from related tasks to improve the primary model's generalization performance. It achieves this goal by training multiple tasks in parallel while sharing representations, where the training signals from the auxiliary tasks can help improve the performance of the primary task. Multi-task learning has been applied to a wide range of natural language processing problems (Luong et al., 2015; Pasunuru and Bansal, 2017; Hashimoto et al., 2017; Ruder et al., 2017b; Kaiser et al., 2017; McCann et al., 2018). Despite its impressive performance, the design of a multi-task learning system is non-trivial. In the context of improving the primary task's performance using knowledge from other auxiliary tasks (Luong et al., 2015; Pasunuru and Bansal, 2017), two major challenges include selecting the most relevant auxiliary tasks and also learning the balanced mixing ratio for synergized training of these tasks. One can achieve this via manual intuition or hyper-parameter tuning over all combinatorial task choices, but this introduces human inductive bias or is not scalable when the number of candidate auxiliary tasks is considerable. To this end, we present AUTOSEM, a two-stage Bayesian optimization pipeline to this problem.

In our AUTOSEM framework[1], the first stage addresses automatic task selection from a pool of auxiliary tasks. For this, we use a non-stationary multi-armed bandit controller (MAB) (Bubeck et al., 2012; Raj and Kalyani, 2017) that dynamically alternates among task choices within the training loop, and eventually returns estimates of the utility of each task w.r.t. the primary task. We model the utility of each task as a Beta distribution, whose expected value can be interpreted as the probability of each task making a non-negative contribution to the training performance of the primary task. Further, we model the observations as Bernoulli variables so that the posterior distribution is also Beta-distributed. We use Thompson sampling (Chapelle and Li, 2011; Russo et al., 2018) to trade off exploitation and exploration.

The second stage then takes the auxiliary tasks

---

[1] We make all our code and models publicly available at: https://github.com/HanGuo97/AutoSeM

selected in the first stage and automatically learns the training mixing ratio of these tasks, through the framework of Bayesian optimization, by modeling the performance of each mixing ratio as a sample from a Gaussian Process (GP) to sequentially search for the optimal values (Rasmussen, 2004; Snoek et al., 2012). For the covariance function in the GP, we use the Matern kernel which is parameterized by a smoothness hyperparameter so as to control the level of differentiability of the samples from GP. Further, following Hoffman et al. (2011), we use a portfolio of optimistic and improvement-based policies as acquisition functions (Shahriari et al., 2016) for selecting the next sample point from the GP search space.

We conduct several experiments on the GLUE natural language understanding benchmark (Wang et al., 2018), where we choose each of RTE, MRPC, QNLI, CoLA, and SST-2 as the primary task, and treat the rest of the classification tasks from the GLUE benchmark as candidate auxiliary tasks. Results show that our AUTOSEM framework can successfully find useful auxiliary tasks and automatically learn their mixing ratio, achieving significant performance boosts on top of strong baselines for several primary tasks, e.g., 5.2% improvement on QNLI, 4.7% improvement on RTE, and 2.8%/0.8% improvement on MRPC.

We also ablate the usefulness of our two stages of auxiliary task selection and automatic mixing ratio learning. The first ablation removes the task selection stage and instead directly performs the second GP mixing ratio learning stage on all auxiliary tasks. The second ablation performs the task selection stage (with multi-armed bandit) but replaces the second stage Gaussian Process with manual tuning on the selected tasks. Our 2-stage model performs better than both these ablations, showing that both of our stages are crucial. Further, we also discuss the learned auxiliary task choices in terms of their intuitive relevance w.r.t. the corresponding primary task.

## 2 Related Work

Multi-task learning (Caruana, 1998), known for improving the generalization performance of a task with auxiliary tasks, has successfully been applied to many domains of machine learning, including natural language processing (Collobert and Weston, 2008; Girshick, 2015; Luong et al., 2015; Pasunuru and Bansal, 2017; Pa-

sunuru et al., 2017), computer vision (Misra et al., 2016; Kendall et al., 2017; Dai et al., 2016), and reinforcement learning (Teh et al., 2017; Parisotto et al., 2015; Jaderberg et al., 2016). Although there are many variants of multi-task learning (Ruder et al., 2017b; Hashimoto et al., 2017; Luong et al., 2015; McCann et al., 2018), our goal is to improve the performance of a primary task using a set of relevant auxiliary tasks, where different tasks share some common model parameters with alternating mini-batches optimization, similar to Luong et al. (2015).

To address the problem of automatic shared parameter selection, Ruder et al. (2017a) automatically learned the latent multi-task sharing architecture, and Xiao et al. (2018) used a gate mechanism that filters the feature flows between tasks. On the problem of identifying task relatedness, Ben-David and Schuller (2003) provided a formal framework for task relatedness and derived generalization error bounds for learning of multiple tasks. Bingel and Søgaard (2017) explored task relatedness via exhaustively experimenting with all possible two task tuples in a non-automated multi-task setup. Other related works explored data selection, where the goal is to select or reorder the examples from one or more domains (usually in a single task) to either improve the training efficiency or enable better transfer learning. These approaches have been applied in machine translation (van der Wees et al., 2017), language models (Moore and Lewis, 2010; Duh et al., 2013), dependency parsing (Søgaard, 2011), etc. In particular, Ruder and Plank (2017) used Bayesian optimization to select relevant training instances for transfer learning, and Tsvetkov et al. (2016) applied it to learn a curriculum for training word embeddings via reordering data. Graves et al. (2017) used the bandit approach (Exp3.S algorithm) in the context of automated curriculum learning, but in our work, we have two stages with each stage addressing a different problem (automatic task selection and learning of the training mixing ratio). Recently, Sharma and Ravindran (2017) used multi-armed bandits (MAB) to learn the choice of hard vs. easy domain data selection as input feed for the model. Guo et al. (2018) used MAB to effectively switch across tasks in a dynamic multi-task learning setup. In our work, we use MAB with Thompson Sampling for the novel paradigm of automatic auxiliary task selection;

and next, we use a Matern-kernel Gaussian Process to automatically learn an exact (static) mixing ratio (i.e., relatedness ratio) for the small number of selected tasks.

Many control problems can be cast as a multi-armed bandits problem, where the goal of the agent is to select the arm/action from one of the $N$ choices that minimizes the regrets (Bubeck et al., 2012). One problem in bandits learning is the trade-off between exploration and exploitation, where the agent needs to make a decision between taking the action that yields the best payoff on current estimates or exploring new actions whose payoffs are not yet certain. Many previous works have explored various exploration and exploitation strategies to minimize regret, including Boltzmann exploration (Kaelbling et al., 1996), adversarial bandits (Auer et al., 2002b), UCB (Auer et al., 2002a), and information gain using variational approaches (Houthooft et al., 2016). In this work, for task selection, we use Thompson Sampling (Russo et al., 2018; Chapelle and Li, 2011), an algorithm for sequential decision making problems, which addresses a broad range of problems in a computationally efficient manner and is therefore enjoying wide use.

Gaussian Process (GP) is a non-parametric Bayesian approach, and it can capture a wide variety of underlying functions or relations between inputs and outputs by taking advantage of the full information provided by the history of observations and is thus very data-efficient (Rasmussen, 2004; Shahriari et al., 2016; Schulz et al., 2018). Gaussian Processes have been widely used as a black-box optimizer and hyper-parameter optimization (Snoek et al., 2012; Brochu et al., 2010; Knudde et al., 2017; Cully et al., 2018; Swersky et al., 2013; Golovin et al., 2017). In our work, we use Gaussian Process for automatic learning of the multi-task mixing ratio in our stage-2 among the selected tasks from stage-1.

## 3   Models

We will first introduce our baseline model and its integration for multiple classification tasks in a multi-task learning (MTL) setup. Next, we will introduce our AUTOSEM framework, an automatic way of selecting auxiliary tasks and learning their optimal training mixing ratio w.r.t. the primary task, via a Beta-Bernoulli bandit with Thompson Sampling and a Gaussian Process framework.
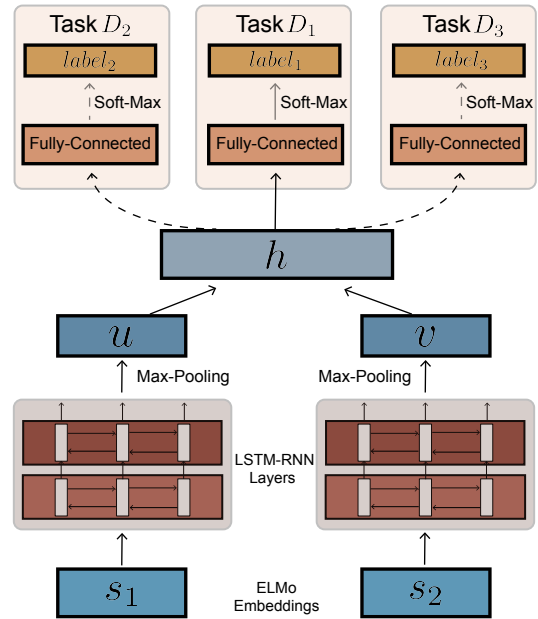


Figure 1: Overview of our baseline model where we use different projection layers for each task during MTL, while sharing rest of the model parameters.

### 3.1   Bi-Text Classification Model

Let $s_1$ and $s_2$ be the input sentence pair in our classification task, where we encode these sentences via bidirectional LSTM-RNN, similar to that of Conneau et al. (2017). Next, we do max-pooling on the output hidden states of both encoders where $u$ and $v$ are the outputs from the max-pooing layer for $s_1$ and $s_2$ respectively. Later, we map these two representations ($u$ and $v$) into a single rich dense representation vector $h$:

$$h = [u; v; u \star v; |u - v|] \tag{1}$$

where $[;]$ represents the concatenation and $u \star v$ represents the element-wise multiplication of $u$ and $v$. We project this final representation $h$ to label space to classify the given sentence pair (see Fig. 1). We also use ELMo (Peters et al., 2018) representations for word embeddings in our model. For this, we extract the three ELMo layer representations for each of the sentence pair and use their weighted sum as the ELMo output representation, where the weights are trainable.

### 3.2   Multi-Task Learning

In this work, we focus on improving a task (primary task) by allowing it to share parameters with related auxiliary tasks via multi-task learning (MTL). Let $\{D_1, ..., D_N\}$ be a set of $N$ tasks, where we set $D_1$ to be the primary task and the rest of them as auxiliary tasks. We can extend
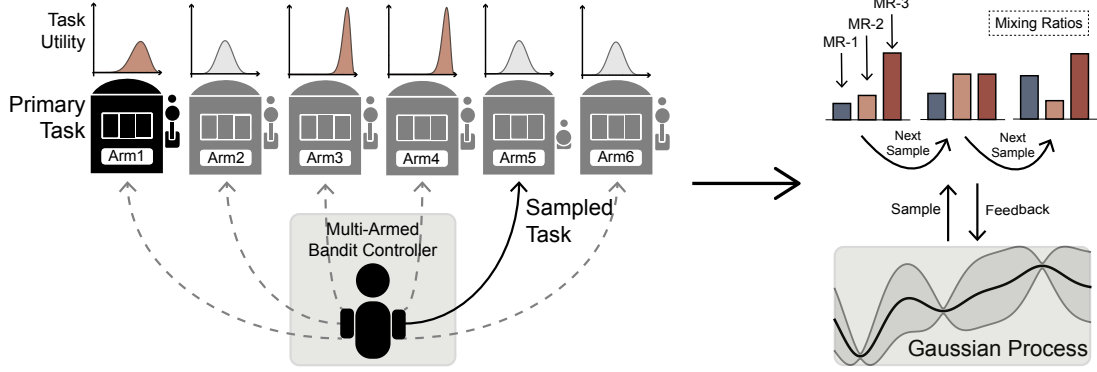
Figure 2: Overview of our AUTOSEM framework. **Left**: the multi-armed bandit controller used for task selection, where each arm represents a candidate auxiliary task. The agent iteratively pulls an arm, observes a reward, updates its estimates of the arm parameters, and samples the next arm. **Right**: the Gaussian Process controller used for automatic mixing ratio (MR) learning. The GP controller sequentially makes a choice of mixing ratio, observes a reward, updates its estimates, and selects the next mixing ratio to try, based on the full history of past observations.

our single-task learning baseline (see Sec. 3.1) into multi-task learning model by augmenting the model with $N$ projection layers while sharing the rest of the model parameters across these $N$ tasks (see Fig. 1). We employ MTL training of these tasks in alternate mini-batches based on a mixing ratio $\eta_1:\eta_2:..\eta_N$, similar to previous work (Luong et al., 2015), where we optimize $\eta_i$ mini-batches of task $i$ and go to the next task.

In MTL, choosing the appropriate auxiliary tasks and properly tuning the mixing ratio can be important for the performance of multi-task models. The naive way of trying all combinations of task selections is hardly tractable. To solve this issue, we propose AUTOSEM, a two-stage pipeline in the next section. In the first stage, we automatically find the relevant auxiliary tasks (out of the given $N - 1$ options) which improve the performance of the primary task. After finding the relevant auxiliary tasks, in the second stage, we take these selected tasks along with the primary task and automatically learn their training mixing ratio.

### 3.3 Automatic Task Selection: Multi-Armed Bandit with Thompson Sampling

Tuning the mixing ratio for $N$ tasks in MTL becomes exponentially harder as the number of auxiliary tasks grows very large. However, in most circumstances, only a small number of these auxiliary tasks are useful for improving the primary task at hand. Manually searching for this optimal choice of relevant tasks is intractable. Hence, in this work, we present a method for automatic task selection via multi-armed bandits with Thompson Sampling (see the left side of Fig. 2).

Let $\{a_1, ..., a_N\}$ represent the set of $N$ arms (corresponding to the set of tasks $\{D_1, ..., D_N\}$) of the bandit controller in our multi-task setting, where the controller selects a sequence of actions/arms over the current training trajectory to maximize the expected future payoff. At each round $t_b$, the controller selects an arm based on the noisy value estimates and observes rewards $r_{t_b}$ for the selected arm. Let $\theta_k \in [0, 1]$ be the utility (usefulness) of task $k$. Initially, the agent begins with an independent prior belief over $\theta_k$. We take these priors to be Beta-distributed with parameters $\alpha_k$ and $\beta_k$, and the prior probability density function of $\theta_k$ is:

$$p(\theta_k) = \frac{\Gamma(\alpha_k + \beta_k)}{\Gamma(\alpha_k)\Gamma(\beta_k)} \theta_k^{\alpha_k - 1} (1 - \theta_k)^{\beta_k - 1} \quad (2)$$

where $\Gamma$ denotes the gamma function. We formulate the reward $r_{t_b} \in \{0, 1\}$ at round $t_b$ as a Bernoulli variable, where an action $k$ produces a reward of 1 with a chance of $\theta_k$ and a reward of 0 with a chance of $1 - \theta_k$. The true utility of task $k$, i.e., $\theta_k$, is unknown, and may or may not change over time (based on stationary vs. non-stationary of task utility). We define the reward as whether sampling the task $k$ improves (or maintains) the validation metric of the primary task,

$$r_{t_b} = \begin{cases} 1, & \text{if } R_{t_b} \geq R_{t_b-1} \\ 0, & \text{otherwise} \end{cases} \quad (3)$$

where $R_{t_b}$ represents the validation performance of the primary task at time $t_b$. With our reward setup above, the utility of each task ($\theta_k$) can be intuitively interpreted as the probability

that multi-task learning with task $k$ can improve (or maintain) the performance of the primary task. The conjugacy properties of the Beta distribution assert that the posterior distribution is also Beta with parameters that can be updated using a simple Bayes rule, which is defined as follows (Russo et al., 2018),

$$p(\theta_k|r) \propto \text{Bern}_\theta(r)\text{Beta}_{\alpha,\beta}(\theta_k)$$
$$\propto \text{Beta}_{\alpha+r,\beta+1-r}(\theta_k) \quad (4)$$

$$(\alpha_k, \beta_k) = \begin{cases} (\alpha_k, \beta_k), & \text{if } x_{t_b}^s \neq k \\ (\alpha_k, \beta_k) + (r_{t_b}, 1 - r_{t_b}), & \text{if } x_{t_b}^s = k \end{cases} \quad (5)$$

where $x_{t_b}^s$ is the sampled task at round $t_b$. Finally, at the end of the training, we calculate the expected value of each arm as follows:

$$\mathbb{E}_p[\theta_k] = \frac{\alpha_k}{\alpha_k + \beta_k} \quad (6)$$

Here, the expectation measures the probability of improving (or maintaining) the primary task by sampling this task. To decide the next action to take, we apply Thompson Sampling (Russo et al., 2018; Chapelle and Li, 2011) to trade off exploitation (maximizing immediate performance) and exploration (investing to accumulate new information that might improve performance in the future). In Thompson Sampling (Russo et al., 2018), instead of taking action $k$ that maximizes the expectation (i.e., $\arg\max_k \mathbb{E}_p[\theta_k]$), we randomly sample the primary task improvement probability $\hat\theta_k$ from the posterior distribution $\hat\theta_k \sim p(\theta_k)$, and take the action $k$ that maximizes the sampled primary task improvement probability, i.e., $\arg\max_k \hat\theta_k$. At the end of the training, the task selection can proceed either via a threshold on the expectation, or take the top-$K$ tasks, and run stage-2 using the selected task subset as auxiliary tasks (details in Sec. 3.4).

**Stronger Prior for Primary Task** Note that at the beginning of training, model performance is usually guaranteed to improve from the initial random choices. This causes issues in updating arm values because less useful tasks will be given high arm values when they happen to be sampled at the beginning. To resolve this issue, we initially set a slightly stronger prior/arm-value in favor of the arm corresponding to the primary task. Intuitively, the bandit will then sample the primary model more often at the beginning, and then start exploring auxiliary tasks when the primary model's

---

**Algorithm 1** BernThompson($N, \alpha, \beta, \gamma, \alpha_0, \beta_0$)

1: **for** $t_b = 1, 2, \dots$ **do**
2:      # sample model:
3:      **for** $k = 1, \dots, N$ **do**
4:          Sample $\hat\theta_k \sim \text{Beta}(\alpha_k, \beta_k)$
5:      **end for**
6:      # select and apply action:
7:      $x_{t_b}^s \leftarrow \arg\max_k \hat\theta_k$
8:      Apply $x_{t_b}^s$ and observe $r_{t_b}$
9:      # non-stationarity
10:     **for** $k = 1, \dots, N$ **do**
11:        $\hat\alpha_k = (1 - \gamma)\alpha_k + \gamma\alpha_0$
12:        $\hat\beta_k = (1 - \gamma)\beta_k + \gamma\beta_0$
13:        **if** $k \neq x_{t_b}^s$ **then**
14:           $(\alpha_k, \beta_k) \leftarrow (\hat\alpha_k, \hat\beta_k)$
15:        **else**
16:           $(\alpha_k, \beta_k) \leftarrow (\hat\alpha_k, \hat\beta_k) + (r_{t_b}, 1 - r_{t_b})$
17:        **end if**
18:     **end for**
19: **end for**

---

performance stabilizes (as the arm value of the primary model will start decreasing because sampling it in later rounds produces smaller additional improvements).

**Non-Stationary Multi-Armed Bandit** Also note that the intrinsic usefulness of each task varies throughout the training (e.g., the primary task might be more important at the beginning, but not necessarily at the end), and thus the agent faces a non-stationary system. In such cases, the agent should always be encouraged to explore in order to track changes as the system drifts. One simple approach to inject non-stationarity is to discount the relevance of previous observations. Thus we introduce a tunable decay ratio $\gamma$, and modify Eq. 3.3 as follows:

$$(\alpha_k, \beta_k) = \begin{cases} (\hat\alpha_k, \hat\beta_k), & \text{if } k \neq x_{t_b}^s \\ (\hat\alpha_k, \hat\beta_k) + (r_{t_b}, 1 - r_{t_b}), & \text{if } k = x_{t_b}^s \end{cases} \quad (7)$$

where $\hat\alpha_k = (1-\gamma)\alpha_k + \gamma\alpha_0$ and $\hat\beta_k = (1-\gamma)\beta_k + \gamma\beta_0$, and $\gamma$ controls how quickly uncertainty is injected into the system ($\alpha_0, \beta_0$ are parameters of the prior). Algorithm 1 presents the Thompson Sampling algorithm with a Beta-Bernoulli MAB.

### 3.4 Automatic Mixing Ratio Learning via Gaussian Process

The right side of Fig. 2 illustrates our Gaussian Process controller for automatic learning of the MTL training mixing ratio (see definition in Sec. 3.2). Given the selected auxiliary tasks from the previous section, the next step is to find a proper mixing ratio of training these selected tasks

along with the primary task.[2] Manual tuning of this mixing ratio via a large grid search over the hyperparameter values is very time and compute expensive (even when the number of selected auxiliary tasks is small, e.g., 2 or 3). Thus, in our second stage, we instead apply a non-parametric Bayesian approach to search for the approximately-optimal mixing ratio. In particular, we use a 'Gaussian Process' to sequentially search for the mixing ratio by trading off exploitation and exploration automatically. Next, we describe our Gaussian Process approach in detail.

A Gaussian Process (Rasmussen, 2004; Snoek et al., 2012; Shahriari et al., 2016), $GP(\mu_0, k)$, is a non-parametric model that is fully characterized by a mean function $\mu_0 : \mathcal{X} \mapsto \mathbb{R}$ and a positive-definite kernel or covariance function $k : \mathcal{X} \times \mathcal{X} \mapsto \mathbb{R}$. Let $\boldsymbol{x}_1, \boldsymbol{x}_2, ..., \boldsymbol{x}_n$ denote any finite collections of $n$ points, where each $\boldsymbol{x}_i$ represents a choice of the mixing ratio (i.e., the ratio $\eta_1:\eta_2:..\eta_N$ described in Sec. 3.2), and $f_i = f(\boldsymbol{x}_i)$ is the (unknown) function values evaluated at $\boldsymbol{x}_i$ (true performance of the model given the selected mixing ratio). Let $y_1, y_2, ..., y_n$ be the corresponding noisy observations (the validation performance at the end of training). In the context of GP Regression (GPR), $\boldsymbol{f} = \{f_1, ..., f_n\}$ are assumed to be jointly Gaussian (Rasmussen, 2004), i.e., $\boldsymbol{f}|\boldsymbol{X} \sim \mathcal{N}(\boldsymbol{m}, \boldsymbol{K})$, where, $\boldsymbol{m}_i = \mu_0(\boldsymbol{x}_i)$ is the mean vector, and $\boldsymbol{K}_{i,j} = k(\boldsymbol{x}_i, \boldsymbol{x}_j)$ is the covariance matrix. Then the noisy observations $\boldsymbol{y} = y_1, ..., y_n$ are normally distributed around $\boldsymbol{f}$ as follows: $\boldsymbol{y}|\boldsymbol{f} \sim \mathcal{N}(\boldsymbol{f}, \sigma^2 \boldsymbol{I})$.

Given $\mathcal{D} = (\boldsymbol{x}_1, y_1), ..., (\boldsymbol{x}_{n_0}, y_{n_0})$, the set of random initial observations, where $\boldsymbol{x}_i$ represents a mixing ratio and $y_i$ represents the corresponding model's validation performance. Next, we model the GP based on these initial observations as described above. We sample a next point $\boldsymbol{x}_{n_0+1}$ (a mixing ratio in our case) from this GP and get its corresponding model performance $y_{n_0+1}$, and update the GP again by now considering the $n_0 + 1$ points (Rasmussen, 2004). We continue this process for a fixed number of steps. Next, we will discuss how we perform the sampling (based on acquisition functions) and the kernels used for calculating the covariance.

**Acquisition Functions** Here, we describe the acquisition functions for deciding where to sample next. While one could select the points that maximize the mean function, this does not always lead to the best outcome (Hoffman et al., 2011). Since we also have the variance of the estimates along with the mean value of each point $\boldsymbol{x}_i$, we can incorporate this information into the optimization. In this work, we use the GP-Hedge approach (Hoffman et al., 2011; Auer et al., 1995), which probabilistically chooses one of three acquisition functions: probability of improvement, expected improvement, and upper confidence bound. Probability of improvement acquisition functions measure the probability that the sampled mixing ratio $\boldsymbol{x}_i$ leads to an improvement upon the best observed value so far ($\tau$), $\mathbb{P}(f(\boldsymbol{x}_i) > \tau)$. Expected improvement additionally incorporates the amount of improvement, $\mathbb{E}[(f(\boldsymbol{x}_i) - \tau)\mathbb{I}(f(\boldsymbol{x}_i) > \tau)]$. The Gaussian Process upper confidence bound (GP-UCB) algorithm measures the optimistic performance upper bound of the sampled mixing ratio (Srinivas et al., 2009), $\mu_i(\boldsymbol{x}_i) + \lambda\sigma_i(\boldsymbol{x}_i)$, for some hyper-parameter $\lambda$.

**Matern Kernel** The covariance function (or kernel) defines the nearness or similarity of two points in the Gaussian Process. Here, we use the automatic relevance determination (ARD) Matern kernel (Rasmussen, 2004), which is parameterized by $\nu > 0$ that controls the level of smoothness. In particular, samples from a GP with such a kernel are differentiable $\lfloor \nu - 1 \rfloor$ times. When $\nu$ is half-integer (i.e. $\nu = p + 1/2$ for non-negative integer $p$), the covariance function is a product of an exponential and a polynomial of order $p$. In the context of machine learning, usual choices of $\nu$ include $3/2$ and $5/2$ (Shahriari et al., 2016).

## 4 Experiment Setup

**Datasets**: We evaluate our models on several datasets from the GLUE benchmark (Wang et al., 2018): RTE, QNLI, MRPC, SST-2, and CoLA. For all these datasets, we use the standard splits provided by Wang et al. (2018). For dataset details, we refer the reader to the GLUE paper.[3]

---

[2] Note that ideally Gaussian Process can also learn to set the mixing ratio of less important tasks to zero, hence allowing it to essentially also perform the task selection step. However, in practice, first applying our task selection Thompson-Sampling model (Sec. 3.3) allows GP to more efficiently search the mixing ratio space for the small number of filtered auxiliary tasks, as shown in results of Sec. 6.1.

[3] We did not include the remaining tasks as primary tasks, because STS-B is a regression task; MNLI is a very large dataset and does not benefit much from MTL with other tasks in the GLUE benchmark; and QQP and WNLI have dev/test discrepancies and adversarial label issues as per the GLUE

| Models | RTE | MRPC | QNLI | CoLA | SST-2 |
|---|---|---|---|---|---|
| BiLSTM+ELMo (Single-Task) (Wang et al., 2018) | 50.1 | 69.0/80.8 | 69.4 | **35.0** | 90.2 |
| BiLSTM+ELMo (Multi-Task) (Wang et al., 2018) | 55.7 | 76.2/83.5 | 66.7 | 27.5 | 89.6 |
| Our Baseline | 54.0 | 75.7/83.7 | 74.0 | 30.8 | 91.3 |
| Our AUTOSEM | **58.7** | **78.5/84.5** | **79.2** | 32.9 | **91.8** |

Table 1: Test GLUE results of previous work, our baseline, and our AUTOSEM MTL framework. We report accuracy and F1 for MRPC, Matthews correlation for CoLA, and accuracy for all others.

**Training Details**: We use pre-trained ELMo[4] to obtain sentence representations as inputs to our model (Peters et al., 2018), and the Gaussian Process implementation is based on Scikit-Optimize[5], and we adopt most of the default configurations. We use accuracy as the validation criterion for all tasks. For all of our experiments except QNLI and SST-2, we apply early stopping on the validation performance plateau.[6] The set of candidate auxiliary tasks consists of all 2-sentence classification tasks when the primary task is a classification of two sentences, whereas it consists of all two-sentence and single-sentence classification tasks when the primary task is a classification of a single sentence.[7] Since the utility estimates from the multi-armed bandit controller are noisy, we choose the top two tasks based on expected task utility estimates, and include additional tasks if their utility estimate is above 0.5. All the results reported are the aggregate of the same experiment with two runs (with different random seeds) unless explicitly mentioned.[8] We use a two-layer LSTM-RNN with hidden size of 1024 for RTE and 512 for the rest of the models, and use Adam Optimizer (Kingma and Ba, 2014). The prior parameters of each task in stage-1 are set to be $\alpha_0 = 1$, $\beta_0 = 1$, which are commonly used in other literature. For stage-1, the bandit controller iteratively selects batches of data from different tasks during training to learn the approximate importance of each auxiliary task (Graves et al., 2017). In stage-2 (Gaussian Process), we sequentially draw samples of mixing ratios and evaluate each sample after full training (Snoek et al., 2012). Without much tuning, we used approximately 200 rounds

for the stage-1 bandit-based approach, where each round consist of approximately 10 mini-batches of optimization. For stage-2, we experimented with 15 and 20 as the number of samples to draw and found that 15 samples for MRPC and 20 samples for the rest of the tasks work well. This brings the total computational cost for our two-stage pipeline to be approximately (15+1)x and (20+1)x, where x represents the time taken to run the baseline model for the given task. This is significantly more efficient than a grid-search based manually-tuned mixing ratio setup (which would scale exponentially with the number of tasks).

## 5 Results

### 5.1 Baseline Models

Table 1 shows the results of our baseline and previous works (Wang et al., 2018). We can see that our single-task baseline models achieve stronger performance on almost all tasks in comparison to previous work's single-task models.[9] Next, we present the performance of our AUTOSEM framework on top of these strong baselines.

### 5.2 Multi-Task Models

Table 1 also presents the performance of our AUTOSEM framework-based MTL models. As can be seen, our MTL models improve significantly (see Table 3 for standard deviations) upon their corresponding single-task baselines for all tasks, and achieve strong improvements as compared to the fairly-comparable[9] multi-task results of previous work (Wang et al., 2018).[10] During the task

---

website's FAQ: https://gluebenchmark.com/faq

[4] https://allennlp.org/elmo

[5] https://scikit-optimize.github.io

[6] In our initial experiments, we found early stopping on larger datasets led to sub-optimal performance, and hence we used a pre-specified maximum number of steps instead.

[7] We made this design decision because there are only two single-sentence tasks in GLUE, so we mix them with 2-sentence tasks to allow more auxiliary choices.

[8] We use the average of validation results across runs as the tuning criterion, and use the ensemble of models across runs for reporting the test results.

[9] Note that we do not report previous works which *fine-tune* large external language models for the task (e.g., OpenAI-GPT and BERT), because they are not fairly comparable w.r.t. our models. Similarly, we report the non-attention based best GLUE models (i.e., BiLSTM+ELMo) for a fair comparison to our non-attention baseline. Our approach should ideally scale to large pre-training/fine-tuning models like BERT, given appropriate compute resources.

[10] Note that even though the performance improvement gaps of Wang et al. (2018) (MTL vs. baseline) and our improvements (AUTOSEM vs. our improved baseline) are similar, these are inherently two different setups. Wang et al. (2018) MTL is based on a 'one model for all' setup (Kaiser et al., 2017; McCann et al., 2018), whereas our approach in-

selection stage of our AUTOSEM framework, we observe that MultiNLI is chosen as one of the auxiliary tasks in all of our MTL models. This is intuitive given that MultiNLI contains multiple genres covering diverse aspects of the complexity of language (Conneau et al., 2017). Also, we observe that WNLI is sometimes chosen in the task selection stage; however, it is always dropped (mixing ratio of zero) by the Gaussian Process controller, showing that it is not beneficial to use WNLI as an auxiliary task (intuitive, given its small size). Next, we discuss the improvements on each of the primary tasks and the corresponding auxiliary tasks selected by AUTOSEM framework.

**RTE**: Our AUTOSEM approach achieves stronger results w.r.t. the baseline on RTE (58.7 vs. 54.0). During our task selection stage, we found out that QQP and MultiNLI tasks are important for RTE as auxiliary tasks. For the second stage of automatic mixing ratio learning via Gaussian Process, the model learns that a mixing ratio of 1:5:5 works best to improve the primary task (RTE) using related auxiliary tasks of QQP and MultiNLI.

**MRPC**: AUTOSEM here performs much better than the baseline on MRPC (78.5/84.5 vs. 75.7/83.7). During our task selection stage, we found out that RTE and MultiNLI tasks are important for MRPC as auxiliary tasks. In the second stage, AUTOSEM learned a mixing ratio of 9:1:4 for these three tasks (MRPC:RTE:MultiNLI).

**QNLI**: Again, we achieve substantial improvements with AUTOSEM w.r.t. baseline on QNLI (79.2 vs. 74.0). Our task selection stage learned that WNLI and MultiNLI tasks are best as auxiliary tasks for QNLI. We found that the Gaussian Process further drops WNLI by setting its mixing ratio to zero, and returns 20:0:5 as the best mixing ratio for QNLI:WNLI:MultiNLI.

**CoLA**: We also observe a strong performance improvement on CoLA with our AUTOSEM model w.r.t. our baseline (32.9 vs. 30.8). During our task selection stage, we found out that MultiNLI and WNLI tasks are important for CoLA as auxiliary tasks. In the second stage, GP learns to drop WNLI, and found the mixing ratio of 20:5:0 for CoLA:MultiNLI:WNLI.

**SST-2**: Here also our AUTOSEM approach performs better than the baseline (91.8 vs. 91.3). The task selection stage chooses MultiNLI, MRPC,

| Name | Validation | Test |
|------|-----------|------|
| Baseline | 78.3 | 75.7/83.7 |
| w/o Stage-1 | 80.3 | 76.3/83.8 |
| w/o Stage-2 | 80.3 | 76.7/83.8 |
| Final MTL | 81.2 | 78.5/84.5 |

Table 2: Ablation results on the two stages of our AUTOSEM framework on MRPC.

and WNLI as auxiliary tasks and the stage-2 Gaussian Process model drops MRPC and WNLI by setting their mixing ratio to zero (learns ratio of 13:5:0:0 for SST-2:MultiNLI:MRPC:WNLI).

## 6 Analysis

### 6.1 Ablation on MTL stages

In this section, we examine the usefulness of each stage of our two-stage MTL pipeline.[11]

**Removing Stage-1**: The purpose of the Beta-Bernoulli MAB in stage-1 is to find useful auxiliary tasks for the given primary task. Here, to understand its importance, we remove the task selection part, and instead directly run the Gaussian Process (GP) model on all tasks (see 'w/o Stage-1' row in Table 2). We can see that by removing the task selection stage, the Gaussian Process model can still outperform the baseline, indicating the usefulness of the GP, but the large mixing ratio search space causes the GP to be unable to efficiently find the best mixing ratio setting.

**Removing Stage-2**: Given the selected tasks from stage-1, the goal of the Gaussian Process in stage-2 is to efficiently find the approximately-optimal mixing ratio. To examine its usefulness, we replace the Gaussian Process controller by manually tuning a grid of mixing ratios, where the number of tuning experiments equals to the number of steps used in the Gaussian Process model (for a fair comparison). Table 2 shows the results by removing stage-2. We can see that a grid search over hyper-parameters can improve upon the baseline, indicating the usefulness of stage-1 task selection, but a reasonable-sized fair-comparison grid search (i.e., not exhaustive over all ratio values) is not able to match our stage-2 GP process that leverages prior experimental results to more efficiently find the best setting.

---

terpretably chooses the 2-3 tasks that are most beneficial for the given primary task. Also see Sec. 4 for comparison of training speeds for these two setups.

---

[11]We present this ablation only on MRPC for now, because GP stage-2 takes a lot of time without the task selection stage.
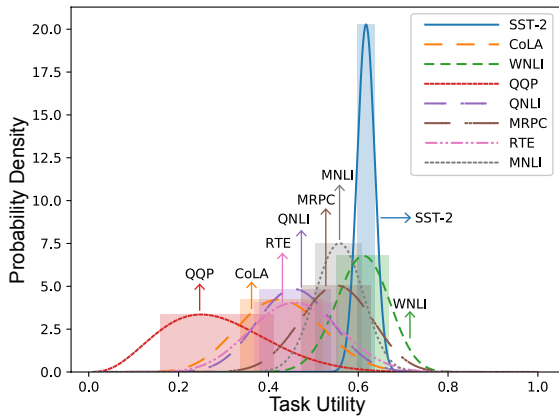
Figure 3: Visualization of task utility estimates from the multi-armed bandit controller on SST-2 (primary task). The x-axis represents the task utility, and the y-axis represents the corresponding probability density. Each curve corresponds to a task and the bar corresponds to their confidence interval.

## 6.2 Stability of MTL Models

In this section, we provide the mean and standard deviation of our baseline and multi-task models (over three runs) on the validation set. Note that the test set is hidden, so we cannot do these studies on it. As seen in Table 3, our multi-task models clearly surpass the performance of baseline models w.r.t. standard deviation gaps, in all tasks.

## 6.3 Visualization of Task Selection

In Fig. 3, we show an example of the task utility estimates from the stage-1 multi-armed bandit controller (Eq. 3.3) on SST-2. The x-axis represents the task utility, and the y-axis represents the probability density over task utility. Each curve represents a task (the blue curve corresponds to the primary task, SST-2, and the rest of the curves correspond to auxiliary tasks), and the width of the bars represents the confidence interval of their estimates. We can see that the bandit controller gives the highest (and most confident) utility estimate for the primary task, which is intuitive given that the primary task should be the most useful task for learning itself. Further, it gives 2-3 tasks moderate utility estimates (the corresponding expected values are around 0.5), and relatively lower utility estimates for the remaining tasks (the corresponding expected values are lower than 0.5).

## 6.4 Educated-Guess Baselines

We additionally experimented with 'educated-guess' baseline models, where MTL is performed using manual intuition mixtures that seem a

| Name | RTE | MRPC | QNLI | CoLA | SST-2 |
|---|---|---|---|---|---|
| BASELINES | | | | | |
| Mean | 58.6 | 78.3 | 74.9 | 74.6 | 91.4 |
| Std | 0.94 | 0.31 | 0.30 | 0.44 | 0.36 |
| MULTI-TASK MODELS | | | | | |
| Mean | 62.0 | 81.1 | 76.0 | 75.7 | 91.8 |
| Std | 0.62 | 0.20 | 0.18 | 0.18 | 0.29 |

Table 3: Validation-set performance mean and standard deviation (based on three runs) of our baselines and Multi-task models in accuracy.

priori sensible.[12]  For example, with MRPC as the primary task, our first educated-guess baseline is to choose other similar paraphrasing-based auxiliary tasks, i.e., QQP in case of GLUE. This MRPC+QQP model achieves 80.8, whereas our AUTOSEM framework chose MRPC+RTE+MultiNLI and achieved 81.2. Furthermore, as our second educated-guess baseline, we added MultiNLI as an auxiliary task (in addition to QQP), since MultiNLI was helpful for all tasks in our MTL experiments. This educated-guess MRPC+QQP+MultiNLI model achieves 80.9 (vs. 81.2 for our AUTOSEM model). This suggests that our AUTOSEM framework (that automatically chose the seemingly less-related RTE task for MRPC) is equal or better than manual intuition based educated-guess models.

## 7 Conclusion

We presented the AUTOSEM framework, a two-stage multi-task learning pipeline, where the first stage automatically selects the relevant auxiliary tasks for the given primary task and the second stage automatically learns their optimal mixing ratio. We showed that AUTOSEM performs better than strong baselines on several GLUE tasks. Further, we ablated the importance of each stage of our AUTOSEM framework and also discussed the intuition of selected auxiliary tasks.

## Acknowledgments

---

[12]These educated-guess models replace our stage-1 automatic auxiliary task section with manual intuition task-mixtures; but we still use our stage-2 Gaussian Process for mixing ratio learning, for fair comparison.

# References

Peter Auer, Nicolo Cesa-Bianchi, and Paul Fischer. 2002a. Finite-time analysis of the multiarmed bandit problem. *Machine learning*, 47(2-3):235–256.

Peter Auer, Nicolo Cesa-Bianchi, Yoav Freund, and Robert E Schapire. 1995. Gambling in a rigged casino: The adversarial multi-armed bandit problem. In *focs*, page 322. IEEE.

Peter Auer, Nicolo Cesa-Bianchi, Yoav Freund, and Robert E Schapire. 2002b. The nonstochastic multi-armed bandit problem. *SIAM journal on computing*, 32(1):48–77.

Shai Ben-David and Reba Schuller. 2003. Exploiting task relatedness for multiple task learning. In *Learning Theory and Kernel Machines*, pages 567–580. Springer.

Joachim Bingel and Anders Søgaard. 2017. Identifying beneficial task relations for multi-task learning in deep neural networks. *arXiv preprint arXiv:1702.08303*.

Eric Brochu, Vlad M Cora, and Nando De Freitas. 2010. A tutorial on bayesian optimization of expensive cost functions, with application to active user modeling and hierarchical reinforcement learning. *arXiv preprint arXiv:1012.2599*.

Sébastien Bubeck, Nicolo Cesa-Bianchi, et al. 2012. Regret analysis of stochastic and nonstochastic multi-armed bandit problems. *Foundations and Trends® in Machine Learning*, 5(1):1–122.

Rich Caruana. 1997. Multitask learning. *Machine learning*, 28(1):41–75.

Rich Caruana. 1998. Multitask learning. In *Learning to learn*, pages 95–133. Springer.

Olivier Chapelle and Lihong Li. 2011. An empirical evaluation of thompson sampling. In *Advances in neural information processing systems*, pages 2249–2257.

Ronan Collobert and Jason Weston. 2008. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the 25th international conference on Machine learning*, pages 160–167. ACM.

Alexis Conneau, Douwe Kiela, Holger Schwenk, Loic Barrault, and Antoine Bordes. 2017. Supervised learning of universal sentence representations from natural language inference data. *arXiv preprint arXiv:1705.02364*.

A. Cully, K. Chatzilygeroudis, F. Allocati, and J.-B. Mouret. 2018. Limbo: A Flexible High-performance Library for Gaussian Processes modeling and Data-Efficient Optimization. *The Journal of Open Source Software*, 3(26):545.

Jifeng Dai, Kaiming He, and Jian Sun. 2016. Instance-aware semantic segmentation via multi-task network cascades. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3150–3158.

Kevin Duh, Graham Neubig, Katsuhito Sudoh, and Hajime Tsukada. 2013. Adaptation data selection using neural language models: Experiments in machine translation. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, volume 2, pages 678–683.

Ross Girshick. 2015. Fast r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 1440–1448.

Daniel Golovin, Benjamin Solnik, Subhodeep Moitra, Greg Kochanski, John Karro, and D Sculley. 2017. Google vizier: A service for black-box optimization. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1487–1495. ACM.

Alex Graves, Marc G Bellemare, Jacob Menick, Remi Munos, and Koray Kavukcuoglu. 2017. Automated curriculum learning for neural networks. *arXiv preprint arXiv:1704.03003*.

Han Guo, Ramakanth Pasunuru, and Mohit Bansal. 2018. Dynamic multi-level multi-task learning for sentence simplification. *arXiv preprint arXiv:1806.07304*.

Kazuma Hashimoto, Caiming Xiong, Yoshimasa Tsuruoka, and Richard Socher. 2017. A joint many-task model: Growing a neural network for multiple nlp tasks. In *EMNLP*.

Matthew D Hoffman, Eric Brochu, and Nando de Freitas. 2011. Portfolio allocation for bayesian optimization. In *UAI*, pages 327–336. Citeseer.

Rein Houthooft, Xi Chen, Yan Duan, John Schulman, Filip De Turck, and Pieter Abbeel. 2016. Vime: Variational information maximizing exploration. In *Advances in Neural Information Processing Systems*, pages 1109–1117.

Max Jaderberg, Volodymyr Mnih, Wojciech Marian Czarnecki, Tom Schaul, Joel Z Leibo, David Silver, and Koray Kavukcuoglu. 2016. Reinforcement learning with unsupervised auxiliary tasks. *arXiv preprint arXiv:1611.05397*.

Leslie Pack Kaelbling, Michael L Littman, and Andrew W Moore. 1996. Reinforcement learning: A survey. *Journal of artificial intelligence research*, 4:237–285.

Lukasz Kaiser, Aidan N Gomez, Noam Shazeer, Ashish Vaswani, Niki Parmar, Llion Jones, and Jakob Uszkoreit. 2017. One model to learn them all. *arXiv preprint arXiv:1706.05137*.

Alex Kendall, Yarin Gal, and Roberto Cipolla. 2017. Multi-task learning using uncertainty to weigh losses for scene geometry and semantics. *arXiv preprint arXiv:1705.07115*, 3.

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980.

Nicolas Knudde, Joachim van der Herten, Tom Dhaene, and Ivo Couckuyt. 2017. GPflowOpt: A Bayesian Optimization Library using TensorFlow. *arXiv preprint – arXiv:1711.03845*.

Minh-Thang Luong, Quoc V Le, Ilya Sutskever, Oriol Vinyals, and Lukasz Kaiser. 2015. Multi-task sequence to sequence learning. *arXiv preprint arXiv:1511.06114*.

Bryan McCann, Nitish Shirish Keskar, Caiming Xiong, and Richard Socher. 2018. The natural language decathlon: Multitask learning as question answering. *arXiv preprint arXiv:1806.08730*.

Ishan Misra, Abhinav Shrivastava, Abhinav Gupta, and Martial Hebert. 2016. Cross-stitch networks for multi-task learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3994–4003.

Robert C Moore and William Lewis. 2010. Intelligent selection of language model training data. In *Proceedings of the ACL 2010 conference short papers*, pages 220–224. Association for Computational Linguistics.

Emilio Parisotto, Jimmy Lei Ba, and Ruslan Salakhutdinov. 2015. Actor-mimic: Deep multitask and transfer reinforcement learning. *arXiv preprint arXiv:1511.06342*.

Ramakanth Pasunuru and Mohit Bansal. 2017. Multi-task video captioning with video and entailment generation. *arXiv preprint arXiv:1704.07489*.

Ramakanth Pasunuru, Han Guo, and Mohit Bansal. 2017. Towards improving abstractive summarization via entailment generation. In *NFiS@EMNLP*.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proc. of NAACL*.

Vishnu Raj and Sheetal Kalyani. 2017. Taming non-stationary bandits: A bayesian approach. *arXiv preprint arXiv:1707.09727*.

Carl Edward Rasmussen. 2004. Gaussian processes in machine learning. In *Advanced lectures on machine learning*, pages 63–71. Springer.

Sebastian Ruder, Joachim Bingel, Isabelle Augenstein, and Anders Søgaard. 2017a. Learning what to share between loosely related tasks. *arXiv preprint arXiv:1705.08142*.

Sebastian Ruder, Joachim Bingel, Isabelle Augenstein, and Anders Søgaard. 2017b. Sluice networks: Learning what to share between loosely related tasks. *arXiv preprint arXiv:1705.08142*.

Sebastian Ruder and Barbara Plank. 2017. Learning to select data for transfer learning with bayesian optimization. *arXiv preprint arXiv:1707.05246*.

Daniel J Russo, Benjamin Van Roy, Abbas Kazerouni, Ian Osband, Zheng Wen, et al. 2018. A tutorial on thompson sampling. *Foundations and Trends® in Machine Learning*, 11(1):1–96.

Eric Schulz, Maarten Speekenbrink, and Andreas Krause. 2018. A tutorial on gaussian process regression: Modelling, exploring, and exploiting functions. *Journal of Mathematical Psychology*, 85:1–16.

Bobak Shahriari, Kevin Swersky, Ziyu Wang, Ryan P Adams, and Nando De Freitas. 2016. Taking the human out of the loop: A review of bayesian optimization. *Proceedings of the IEEE*, 104(1):148–175.

Sahil Sharma and Balaraman Ravindran. 2017. Online multi-task learning using active sampling. In *ICLR*.

Jasper Snoek, Hugo Larochelle, and Ryan P Adams. 2012. Practical bayesian optimization of machine learning algorithms. In *Advances in neural information processing systems*, pages 2951–2959.

Anders Søgaard. 2011. Data point selection for cross-language adaptation of dependency parsers. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies: short papers-Volume 2*, pages 682–686. Association for Computational Linguistics.

Niranjan Srinivas, Andreas Krause, Sham M Kakade, and Matthias Seeger. 2009. Gaussian process optimization in the bandit setting: No regret and experimental design. *arXiv preprint arXiv:0912.3995*.

Kevin Swersky, Jasper Snoek, and Ryan P Adams. 2013. Multi-task bayesian optimization. In *Advances in neural information processing systems*, pages 2004–2012.

Yee Teh, Victor Bapst, Wojciech M Czarnecki, John Quan, James Kirkpatrick, Raia Hadsell, Nicolas Heess, and Razvan Pascanu. 2017. Distral: Robust multitask reinforcement learning. In *Advances in Neural Information Processing Systems*, pages 4496–4506.

Yulia Tsvetkov, Manaal Faruqui, Wang Ling, Brian MacWhinney, and Chris Dyer. 2016. Learning the curriculum with bayesian optimization for task-specific word representation learning. *arXiv preprint arXiv:1605.03852*.

Alex Wang, Amapreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R Bowman. 2018. Glue: A multi-task benchmark and analysis platform for natural language understanding. *arXiv preprint arXiv:1804.07461*.

Marlies van der Wees, Arianna Bisazza, and Christof Monz. 2017. Dynamic data selection for neural machine translation. *arXiv preprint arXiv:1708.00712*.

Liqiang Xiao, Honglun Zhang, and Wenqing Chen. 2018. Gated multi-task network for text classification. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, volume 2, pages 726–731.