# Learning to Few-Shot Learn Across Diverse Natural Language Classification Tasks

**Trapit Bansal**[*†] and **Rishikesh Jha**[*‡] and **Andrew McCallum**[†]
[†]University of Massachusetts, Amherst
[‡]Code for Science and Society
{tbansal,rishikeshjha,mccallum}@cs.umass.edu

## Abstract

Pre-trained transformer models have shown enormous success in improving performance on several downstream tasks. However, fine-tuning on a new task still requires large amounts of task-specific labeled data to achieve good performance. We consider this problem of learning to generalize to new tasks with a few examples as a meta-learning problem. While meta-learning has shown tremendous progress in recent years, its application is still limited to simulated problems or problems with limited diversity across tasks. We develop a novel method, LEOPARD, which enables optimization-based meta-learning across tasks with different number of classes, and evaluate different methods on generalization to diverse NLP classification tasks. LEOPARD is trained with the state-of-the-art transformer architecture and shows better generalization to tasks not seen at all during training, with as few as 4 examples per label. Across 17 NLP tasks, including diverse domains of entity typing, natural language inference, sentiment analysis, and several other text classification tasks, we show that LEOPARD learns better initial parameters for few-shot learning than self-supervised pre-training or multi-task training, outperforming many strong baselines, for example, yielding 14.6% average relative gain in accuracy on unseen tasks with only 4 examples per label.

## 1 Introduction

Learning to learn (Schmidhuber, 1987; Bengio et al., 1992; Thrun and Pratt, 2012) from limited supervision is an important problem with widespread application in areas where obtaining labeled data for training large models can be difficult or expensive. We consider this problem of *learning in $k$-shots* for natural language processing (NLP) tasks, that is, given $k$ labeled examples of a new NLP task learn to efficiently solve the new task. Recently, self-supervised pre-training of transformer models using language modeling objectives (Devlin et al., 2018; Radford et al., 2019; Yang et al., 2019) has achieved tremendous success in learning general-purpose parameters which are useful for a variety of downstream NLP tasks. While pre-training is beneficial, it is not optimized for fine-tuning with limited supervision and such models still require large amounts of task-specific data for fine-tuning, in order to achieve good performance (Yogatama et al., 2019).

On the other hand, meta-learning methods have been proposed as effective solutions for few-shot learning. Existing applications of such meta-learning methods have shown improved performance in few-shot learning for vision tasks such as learning to classify new image classes within a similar dataset. However, these applications are often limited to simulated datasets where each classification label is considered a task. Moreover, their application in NLP has followed a similar trend (Han et al., 2018; Yu et al., 2018; Guo et al., 2018; Mi et al., 2019; Geng et al., 2019). Since the input space of natural language is shared across all NLP tasks, it is possible that a meta-learning approach generalizes to unseen tasks. We thus move beyond simulated tasks to investigate meta-learning performance on generalization outside the training tasks, and focus on a diverse task-set with different number of labels across tasks.

[*]Equal Contribution. Correspondence: tbansal@cs.umass.edu





*Proceedings of the 28th International Conference on Computational Linguistics*, pages 5108–5123
Barcelona, Spain (Online), December 8-13, 2020

Model agnostic meta-learning (MAML) (Finn et al., 2017) is an optimization-based approach to meta-learning which is agnostic to the model architecture and task specification. Hence, it is an ideal candidate for learning to learn from diverse tasks. However, it requires sharing model parameters, including softmax classification layers across tasks and learns a single initialization point across tasks. This poses a barrier for learning across diverse tasks, where different tasks can have potentially disjoint label spaces. Contrary to this, multi-task learning (Caruana, 1997) naturally handles disjoint label sets, while still benefiting from sharing statistical strength across tasks. However, to solve a new task, multi-task learning would require training a new classification layer for the task. On the other hand, metric-based approaches, such as prototypical networks (Vinyals et al., 2016; Snell et al., 2017), being non-parametric in nature can handle varied number of classes. However, as the number of labeled examples increase, these methods do not adapt to leverage larger data and their performance can lag behind optimization-based methods.

We address these concerns and make the following contributions: (1) we introduce a MAML-based meta-learning method, **LEOPARD**[1], which is coupled with a parameter generator that learns to generate *task-dependent* initial softmax classification parameters for any given task and enables meta-learning across tasks with disjoint label spaces; (2) we train LEOPARD with a transformer model, BERT (Devlin et al., 2018), as the underlying neural architecture, and show that it is possible to learn better initialization parameters for few-shot learning than that obtained from just self-supervised pre-training or pre-training followed by multi-task learning; (3) we evaluate on generalization, with a few-examples, to NLP tasks not seen during training or to new domains of seen tasks, including *entity typing, natural language inference, sentiment classification, and various other text classification tasks*; (4) we study how meta-learning, multi-task learning and fine-tuning perform for few-shot learning of completely new tasks, analyze merits/demerits of parameter efficient meta-training, and study how various train tasks affect performance on target tasks. To the best of our knowledge, this is the first application of meta-learning in NLP which evaluates on test tasks which are significantly different than training tasks and goes beyond simulated classification tasks or domain-adaptation tasks (where train and test tasks are similar but from different domains).

## 2 Background

In meta-learning, we consider a meta goal of learning across multiple tasks and assume a distribution over tasks $T_i \sim P(\mathcal{T})$. We follow the episodic learning framework of Vinyals et al. (2016) which minimizes train-test mismatch for few-shot learning. We are given a set of $M$ training tasks $\{T_1, \ldots, T_M\}$, where each task instance potentially has a large amount of training data. In order to simulate $k$-shot learning during training, in each episode (i.e. a training step) a task $T_i$ is sampled with a training set $\mathcal{D}_i^{tr} \sim T_i$, consisting of only $k$ examples (per label) of the task and a validation set $\mathcal{D}_i^{val} \sim T_i$, containing several other examples of the same task. The model $f$ is trained on $\mathcal{D}_i^{tr}$ using the task loss $\mathcal{L}_i$, and then evaluated on $\mathcal{D}_i^{val}$. The loss on $\mathcal{D}_i^{val}$ is then used to adjust the model parameters. Here the validation error of the tasks serves as the training error for the meta-learning process. At the end of training, the model is evaluated on a new task $T_{M+1} \sim P(\mathcal{T})$, where again the train set of $T_{M+1}$ contains only $k$ examples per label, and the model can use its learning procedure to adapt to the task $T_{M+1}$ using the train set. We next discuss model-agnostic meta-learning which is pertinent to our work.

**Model-Agnostic Meta-Learning (MAML)** (Finn et al., 2017) is an approach to optimization-based meta-learning where the goal is to find a good initial point for model parameters $\theta$, which through few steps of gradient descent, can be adapted to yield good performance on a new task. Learning in MAML consists of an *inner loop*, which applies gradient-based learning on the task-specific objective, and an *outer-loop* which refines the initial point across tasks in order to enable fast learning. Given a task $T_i$ with training datasets $\mathcal{D}_i^{tr}$ sampled during an episode, MAML's inner loop adapts the parameters $\theta$ as:

$$\theta_i' = \theta - \alpha \nabla_\theta \mathcal{L}_i(\theta, \mathcal{D}_i^{tr}) \tag{1}$$

Typically, more than one step of gradient update are applied sequentially. The learning-rate $\alpha$ can also be meta-learned in the outer-loop (Li et al., 2017). The parameters $\theta$ are then trained by back-propagating

[1] **L**earning to g**e**nerate s**o**ftmax **pa**rameters fo**r** **d**iverse classification

through the inner-loop adaptation, with the meta-objective of minimizing the error across respective task validation sets $\mathcal{D}_i^{val}$:

$$\theta \leftarrow \theta - \beta \nabla_\theta \sum_{T_i \sim P(\mathcal{T})} \mathcal{L}_i(\theta_i', \mathcal{D}_i^{val}) \tag{2}$$

Note that even though MAML is trained to generate a good initialization point for few-shot adaptation, since the inner-loop employs gradient-based learning, its performance can approach supervised learning in the limit of large data.

## 3 Model

In this section, we describe our proposed method, LEOPARD, for learning new NLP classification tasks with $k$-examples. Fig. 1 shows a high-level description of the model. Our approach builds on the MAML framework and addresses some of its limitations when applied to a diverse set of tasks with different number of classes across tasks. Our model consists of three main components: (1) a shared neural input encoder which generates feature representations useful across tasks; (2) a softmax parameter generator *conditioned on the training dataset* for an $N$-way task, which generates the initial softmax parameters for the task; (3) a MAML-based adaptation method with a distinction between *task-specific parameters*, which are adapted per task, and *task-agnostic* parameters, which are shared across tasks, that can lead to parameter-efficient fine-tuning of large models. Full training algorithm is shown in Alg. 1.

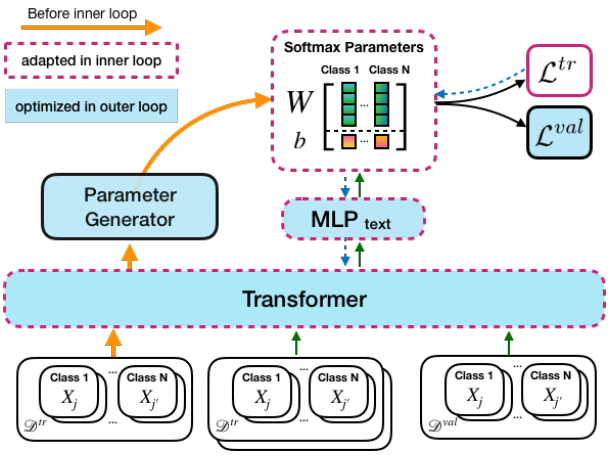


Figure 1: The proposed LEOPARD model. Input is first encoded using the Transformer. The first batch from the support set is passed through the parameter generator which learns a per-class set representation that is used to generate the initial softmax parameters. Subsequently, the support batches are used for adaptation of the generated parameters as well as the encoder parameters. Pink box (dashed) outline shows modules that are adapted in the inner loop, whereas blue boxes are optimized in the outer loop.

### 3.1 Text Encoder

The input consists of natural language sentences, thus our models take sequences of words as input. Note that some tasks require classifying pairs of sentences (such as natural language inference) and phrases in a sentence (such as entity typing), and we discuss how these can also be encoded as a sequence in Section 4.1. We use a Transformer model (Vaswani et al., 2017) as our text encoder which has shown success for many NLP tasks. Concretely, we follow Devlin et al. (2018) and use their BERT-base model architecture. We denote the Transformer model by $f_\theta$, with parameters $\theta = \{\theta_1, \ldots, \theta_{12}\}$ where $\theta_v$ are the parameters of layer $v$. Transformer takes a sequence of words $\mathbf{x}_j = [x_{j1}, \ldots, x_{jt}]$ as input ($t$ being the sequence length), and outputs $d$-dimensional contextualized representations at the final layer of multi-head self-attention. BERT adds a special CLS token (Devlin et al., 2018) to the start of every input, which can be used as a sentence representation. We thus use this as the fixed-dimensional input feature representation of the sentence: $\tilde{x}_j = f_\theta([x_{j1}, \ldots, x_{js}])$.

### 3.2 Generating Softmax Parameters for Task-specific Classification

Existing applications of MAML consider few-shot learning with a fixed $N$, i.e. the number of classes. This limits applicability to multiple types of tasks, each of which would require a different number of classes for classification. To remedy this, we introduce a method to generate *task-dependent* softmax parameters (both linear weights and bias). Given the training data, $\mathcal{D}_i^{tr} = \{(x_j, y_j)\}$, for a task $T_i$ in

**Algorithm 1** LEOPARD

**Require:** set of $M$ training tasks and losses $\{(T_1, L_1), \ldots, (T_M, L_M)\}$, model parameters $\Theta = \{\theta, \psi, \alpha\}$, hyper-parameters $\nu, G, \beta$

Initialize $\theta$ with pre-trained BERT-base;

**while** not converged **do**
  *# sample batch of tasks*
  **for all** $T_i \in T$ **do**
    $\mathcal{D}_i^{tr} \sim T_i$ *# sample a batch of train data*
    $C_i^n \leftarrow \{x_j | y_j = n\}$ *# partition data according to class labels*
    $w_i^n, b_i^n \leftarrow \frac{1}{|C_i^n|} \sum_{x_j \in C_i^n} g_\psi(f_\theta(x_j))$ *# generate softmax parameters*
    $\mathbf{W}_i \leftarrow [w_i^1; \ldots; w_i^{N_i}]$; $\mathbf{b}_i \leftarrow [b_i^1; \ldots; b_i^{N_i}]$
    $\Phi_i^{(0)} \leftarrow \theta_{>\nu} \cup \{\phi, \mathbf{W}_i, \mathbf{b}_i\}$ *# task-specific parameters*
    **for** $s := 0 \ldots G-1$ **do**
      $\mathcal{D}_i^{tr} \sim T_i$ *# sample a batch of train data*
      $\Phi_i^{(s+1)} \leftarrow \Phi_i^{(s)} - \alpha_s \nabla_\Phi \mathcal{L}_i(\{\Theta, \Phi_i\}, \mathcal{D}_i^{tr})$ *# adapt task-specific parameters*
    **end for**
    $\mathcal{D}_i^{val} \sim T_i$ *# sample a batch of validation data*
    $g_i \leftarrow \nabla_\Theta \mathcal{L}_i(\{\Theta, \Phi_i^{(G)}\}, \mathcal{D}_i^{val})$ *# gradient of task-agnostic parameters on validation*
  **end for**
  $\Theta \leftarrow \Theta - \beta \cdot \sum_i g_i$ *# optimize task-agnostic parameters*
**end while**

an episode, we first partition the input into the $N_i$ number of classes for the task (available in $\mathcal{D}_i^{tr}$): $C_i^n = \{x_j | y_j = n\}$, where $n \in [N_i]$. Now, we perform a non-linear projection on the representations of the $x_j$ in each class partition obtained from the text-encoder, and obtain a set representation for class $n$:

$$w_i^n, b_i^n = \frac{1}{|C_i^n|} \sum_{x_j \in C_i^n} g_\psi(f_\theta(\mathbf{x}_j)) \tag{3}$$

where $g_\psi$ is multi-layer perceptron (MLP) with two layers and $tanh$ non-linearities, $w_i^n$ is a $l$-dimensional vector and $b_i^n$ is a scalar. $w_i^n$ and $b_i^n$ are the softmax linear weight and bias, respectively, for the class $n$:

$$\mathbf{W}_i = [w_i^1; \ldots; w_i^{N_i}] \quad \mathbf{b}_i = [b_i^1; \ldots; b_i^{N_i}] \tag{4}$$

Thus, the softmax classification weights $\mathbf{W}_i \in \mathcal{R}^{N_i \times l}$ and bias $\mathbf{b}_i \in \mathcal{R}^{N_i}$ for task $T_i$ are obtained by row-wise concatenation of the per-class weights in equation 3. Note that encoder $g_\psi(\cdot)$ would be shared across tasks in different episodes.

Now, given the softmax parameters, the prediction for a new data-point $\mathbf{x}^*$ is given as:

$$p(y|\mathbf{x}^*) = \text{softmax}\left\{\mathbf{W}_i h_\phi(f_\theta(\mathbf{x}^*)) + \mathbf{b}_i\right\} \tag{5}$$

where $h_\phi(\cdot)$ is another MLP with parameters $\phi$ and output dimension $l$, and the softmax is over the set of classes $N_i$ for the task.

Note that if we use $x^* \in \mathcal{D}_i^{val}$, then the model is a form of a prototypical network (Snell et al., 2017) which uses a learned distance function. However, this would limit the model to not adapt its parameters with increasing data. We next discuss how we learn to adapt using the generated softmax. It is important to note that *we do not introduce any task-specific parameters*, unlike multi-task learning (Caruana, 1997) which will require new softmax layers for each task, and the existing parameters are used to generate a good starting point for softmax parameters across tasks which can then be *adapted* using stochastic gradient (SGD) based learning.

### 3.3 Learning to Adapt Efficiently

Given the task-specific classification loss computed at an episode, MAML takes multiple steps of SGD on the same training set $\mathcal{D}_i^{tr}$, as in equation 1. We apply MAML on the model parameters, including the generated softmax parameters. However, the number of parameters in BERT is substantially high ($\sim$ 110 million) and it can be beneficial to adapt a smaller number of parameters (Houlsby et al., 2019; Zintgraf et al., 2019). We thus separate the set of parameters into *task-specific* and *task-agnostic*. For the transformer parameters for each layer $\{\theta_v\}$, we consider a threshold $\nu$ over layers, and consider $\theta_{\leq\nu} = \{\theta_v | v \leq \nu\}$ to be the parameters for first $\nu$ layers (closest to the input) and the rest of the parameters as $\theta_{>\nu}$. Then we consider $\theta_{\leq\nu}$ and the parameters $\psi$ of the softmax generating function (equation 3) as the set of task-agnostic parameters $\Theta = \theta_{\leq\nu} \cup \{\psi\}$. These task-agnostic parameters $\Theta$ need to generalize to produce good feature representations and good initial point for classification layer *across tasks*. The remaining set of parameters for the higher layers of transformer, the input projection function in 5, and the softmax weights and bias *generated* in equation 4 are considered as the set of task-specific parameters $\Phi_i = \theta_{>\nu} \cup \{\phi, \mathbf{W}_i, \mathbf{b}_i\}$.

The task-specific parameters will be adapted for each task using SGD, as in equation 1. Note that MAML usually does gradient descent steps on the same meta-train batch $\mathcal{D}_i^{tr}$ for a task in an episode. However, since we use $\mathcal{D}_i^{tr}$ to generate the softmax parameters in equation 3, using the same data to also take multiple gradient steps can lead to over-fitting. Thus, we instead sample $G > 1$ meta-train batches in each episode of training, and use the subesequent batches (after the first batch) for adaptation. Task-specific adaptation in the inner loop does $G$ steps of the following update, starting with $\Phi_i^{(0)} \leftarrow \Phi_i$, for $s := 0, \ldots, G-1$:

$$\Phi_i^{(s+1)} = \Phi_i^{(s)} - \alpha_s \, \mathbb{E}_{\mathcal{D}_i^{tr} \sim T_i} \left[ \nabla_\Phi \mathcal{L}_i(\{\Theta, \Phi_i\}, \mathcal{D}_i^{tr}) \right] \tag{6}$$

Note that we only take gradient with respect to the task-specific parameters $\Phi_i$, however the updated parameter is also a function of $\Theta$. After the $G$ steps of adaptation, the final point (which consists of parameters $\Theta$ and $\Phi^G$) is evaluated on the validation set for the task, $\mathcal{D}_i^{val}$, and the task-agnostic parameters $\Theta$ are updated (as in equation 2) to adjust the initial point across tasks. Note that optimization of the task-agnostic parameters requires back-propagating through the inner-loop gradient steps and requires computing higher-order gradients. Finn et al. (2017) proposed using a first-order approximation for computational efficiency. We use this approximation in this work, however we note that the distinction between task-specific and task-agnostic parameters can allow for higher order gradients when there are few task-specific parameters (for example, only the last layer).

**Other Technical Details:** For few-shot learning, learning rate can often be an important hyper-parameter and the above approach can benefit from also learning the learning-rate for adaptation (Li et al., 2017). Instead of scalar inner loop learning rates, it has been shown beneficial to have per-parameter learning rates that are also learned (Li et al., 2017; Antoniou et al., 2018). However, this doubles the number of parameters and can be inefficient. Instead, we learn a per-layer learning rate for the inner loop to allow different transformer layers to adapt at different rates. We apply layer normalization across layers of transformers (Vaswani et al., 2017; Ba et al., 2016) and also adapt their parameters in the inner loop. The number of layers to consider as task-specific, $\nu$, is a hyper-parameter. We initialize the meta-training of LEOPARD from pre-trained BERT model which stabilizes training.

## 4 Experiments

Our experiments evaluate how different methods generalize to new NLP tasks with limited supervision. We focus on sentence-level classification tasks, including natural language inference (NLI) tasks which require classifying pairs of sentences as well as tasks like entity typing which require classifying a phrase in a sentence. We consider 17 target tasks[2]. Main results are in Sec. 4.3.

[2]Code, trained model parameters, and datasets: `https://github.com/iesl/leopard`

### 4.1 Training Tasks

We use the GLUE benchmark tasks (Wang et al., 2018b) for training all the models. Such tasks are considered important for general linguistic intelligence, have lots of supervised data for many tasks and have been useful for transfer learning (Phang et al., 2018; Wang et al., 2018a). We consider the following tasks for training[3]: MNLI (m/mm), SST-2, QNLI, QQP, MRPC, RTE, and the SNLI dataset (Bowman et al., 2015). We use the corresponding validation sets for hyper-parameter tuning and early stopping. For meta-learning methods, we classify between every pair of labels (for tasks with more than 2 labels) which increases the number of tasks and allows for more per-label examples in a batch during training. Moreover, to learn to do phrase-level classification, we modify SST (for all models) which is a phrase-level sentiment classification task by providing a sentence in which the phrase occurs as part of the input. That is, the input is the sentence followed by a separator token (Devlin et al., 2018) followed by the phrase to classify. See Appendix A for more details.

### 4.2 Evaluation and Baselines

Unlike existing methods which evaluate meta-learning models on sampled tasks from a fixed dataset (Vinyals et al., 2016; Finn et al., 2017), we evaluate methods on real NLP datasets by using the entire test sets for the target task after using a sampled $k$-shot training data for fine-tuning. The models parameters are trained on the set of training tasks and are then fine-tuned with $k$ training examples per label for a target test task. The fine-tuned models are then evaluated on the *entire test-set* for the task. We evaluate on $k \in \{4, 8, 16\}$. For each task, for every $k$, we sample 10 training datasets and report the mean and standard deviation, since model performance can be sensitive to the $k$ examples chosen for training. In the few-shot setting it can be unreasonable to assume access to a large validation set (Yu et al., 2018; Kann et al., 2019), thus for the fine-tuning step we tuned the hyper-parameters for all baselines on a held out validation task. We used SciTail, a scientific NLI task, and electronics domain of Amazon sentiment classification task as the validation tasks. We took the hyper-parameters that gave best average performance on validation data of these tasks, for each value of $k$. For LEOPARD, we only tune the number of epochs for fine-tuning, use the learned per-layer learning rates and reuse remaining hyper-parameters (see Appendix C).

We evaluate multiple transfer learning baselines as well as a meta-learning baseline. Note that most existing applications of few-shot learning are tailored towards specific tasks and don't trivially apply to diverse tasks considered here. We evaluate the following methods:

**$\text{BERT}_{\text{base}}$**: We use the cased BERT-base model (Devlin et al., 2018) which is a state-of-the-art transformer (Vaswani et al., 2017) model for NLP. BERT uses language model pre-training followed by supervised fine-tuning on a downstream task. For fine-tuning, we tune all parameters as it performed better on the validation task.

**Multi-task BERT (MT-BERT)**: This is the BERT-base model trained in a multi-task learning setting on the set of training tasks. Our MT-BERT is comparable to the MT-DNN model of Liu et al. (2019) that is trained on the tasks considered here and uses the cased BERT-base as the initialization. We did not use the specialized stochastic answer network for NLI used by MT-DNN. For this model, we tune all the parameters during fine-tuning.

**$\text{MT-BERT}_{\text{softmax}}$**: This is the multi-task BERT model above, where we only tune the softmax layer during fine-tuning.

**Prototypical BERT (Proto-BERT)**: This is the prototypical network method (Snell et al., 2017) that uses BERT-base as the underlying neural model. Following Snell et al. (2017), we used euclidean distance as the distance metric.

All methods are initialized with pre-trained BERT. All parameters of MT-BERT and Proto-BERT are also tuned during training. We don't compare with MAML (Finn et al., 2017) as it does not trivially support varying number of classes, and show in ablations (4.4) that solutions like using zero-initialized initial softmax perform worse.

[3]We exclude WNLI since its training data is small and STS-B task since it is a regression task

**Implementation Details**: Since dataset sizes can be imbalanced, it can affect multi-task and meta-learning performance. Wang et al. (2018a) analyze this in detail for multi-task learning. We explored sampling tasks with uniform probability, proportional to size and proportional to the square-root of the size of the task. For all models, we found the latter to be beneficial. All methods are trained on 4 GPUs to benefit from large batches. Best hyper-parameters, search ranges and data statistics are in Appendix.

### 4.3 Results

We evaluate all the models on 17 target NLP tasks. None of the task data is observed during the training of the models, and the models are fine-tuned on few examples for the target task and then evaluated on the entire test set for the task. For $k$-shot learning of tasks not seen at all during training, we observe, on average, relative gain in accuracy of $14.60\%$, $10.83\%$, and $11.16\%$, for $k = 4, 8, 16$ respectively.

#### 4.3.1 Generalization Beyond Training Tasks

| | $N$ | $k$ | $\text{BERT}_{\text{base}}$ | $\text{MT-BERT}_{\text{softmax}}$ | MT-BERT | Proto-BERT | LEOPARD |
|---|---|---|---|---|---|---|---|
| **Entity Typing** | | | | | | | |
| CoNLL | 4 | 4 | 50.44 ± 08.57 | 52.28 ± 4.06 | **55.63** ± 4.99 | 32.23 ± 5.10 | 54.16 ± 6.32 |
| | | 8 | 50.06 ± 11.30 | 65.34 ± 7.12 | 58.32 ± 3.77 | 34.49 ± 5.15 | **67.38** ± 4.33 |
| | | 16 | 74.47 ± 03.10 | 71.67 ± 3.03 | 71.29 ± 3.30 | 33.75 ± 6.05 | **76.37** ± 3.08 |
| MITR | 8 | 4 | 49.37 ± 4.28 | 45.52 ± 5.90 | **50.49** ± 4.40 | 17.36 ± 2.75 | 49.84 ± 3.31 |
| | | 8 | 49.38 ± 7.76 | 58.19 ± 2.65 | 58.01 ± 3.54 | 18.70 ± 2.38 | **62.99** ± 3.28 |
| | | 16 | 69.24 ± 3.68 | 66.09 ± 2.24 | 66.16 ± 3.46 | 16.41 ± 1.87 | **70.44** ± 2.89 |
| **Text Classification** | | | | | | | |
| Airline | 3 | 4 | 42.76 ± 13.50 | 43.73 ± 7.86 | 46.29 ± 12.26 | 40.27 ± 8.19 | **54.95** ± 11.81 |
| | | 8 | 38.00 ± 17.06 | 52.39 ± 3.97 | 49.81 ± 10.86 | 51.16 ± 7.60 | **61.44** ± 03.90 |
| | | 16 | 58.01 ± 08.23 | 58.79 ± 2.97 | 57.25 ± 09.90 | 48.73 ± 6.79 | **62.15** ± 05.56 |
| Disaster | 2 | 4 | **55.73** ± 10.29 | 52.87 ± 6.16 | 50.61 ± 8.33 | 50.87 ± 1.12 | 51.45 ± 4.25 |
| | | 8 | **56.31** ± 09.57 | 56.08 ± 7.48 | 54.93 ± 7.88 | 51.30 ± 2.30 | 55.96 ± 3.58 |
| | | 16 | 64.52 ± 08.93 | **65.83** ± 4.19 | 60.70 ± 6.05 | 52.76 ± 2.92 | 61.32 ± 2.83 |
| Emotion | 13 | 4 | 09.20 ± 3.22 | 09.41 ± 2.10 | 09.84 ± 2.14 | 09.18 ± 3.14 | **11.71** ± 2.16 |
| | | 8 | 08.21 ± 2.12 | 11.61 ± 2.34 | 11.21 ± 2.11 | 11.18 ± 2.95 | **12.90** ± 1.63 |
| | | 16 | 13.43 ± 2.51 | **13.82** ± 2.02 | 12.75 ± 2.04 | 12.32 ± 3.73 | 13.38 ± 2.20 |
| Political Bias | 2 | 4 | 54.57 ± 5.02 | 54.32 ± 3.90 | 54.66 ± 3.74 | 56.33 ± 4.37 | **60.49** ± 6.66 |
| | | 8 | 56.15 ± 3.75 | 57.36 ± 4.32 | 54.79 ± 4.19 | 58.87 ± 3.79 | **61.74** ± 6.73 |
| | | 16 | 60.96 ± 4.25 | 59.24 ± 4.25 | 60.30 ± 3.26 | 57.01 ± 4.44 | **65.08** ± 2.14 |
| Political Audience | 2 | 4 | 51.89 ± 1.72 | 51.50 ± 2.72 | 51.53 ± 1.80 | 51.47 ± 3.68 | **52.60** ± 3.51 |
| | | 8 | 52.80 ± 2.72 | 53.53 ± 2.26 | **54.34** ± 2.88 | 51.83 ± 3.77 | 54.31 ± 3.95 |
| | | 16 | **58.45** ± 4.98 | 56.37 ± 2.19 | 55.14 ± 4.57 | 53.53 ± 3.25 | 57.71 ± 3.52 |
| Political Message | 9 | 4 | 15.64 ± 2.73 | 13.71 ± 1.10 | 14.49 ± 1.75 | 14.22 ± 1.25 | **15.69** ± 1.57 |
| | | 8 | 13.38 ± 1.74 | 14.33 ± 1.32 | 15.24 ± 2.81 | 15.67 ± 1.96 | **18.02** ± 2.32 |
| | | 16 | **20.67** ± 3.89 | 18.11 ± 1.48 | 19.20 ± 2.20 | 16.49 ± 1.96 | 18.07 ± 2.41 |
| Rating Books | 3 | 4 | 39.42 ± 07.22 | 44.82 ± 9.00 | 38.97 ± 13.27 | 48.44 ± 7.43 | **54.92** ± 6.18 |
| | | 8 | 39.55 ± 10.01 | 51.14 ± 6.78 | 46.77 ± 14.12 | 52.13 ± 4.79 | **59.16** ± 4.13 |
| | | 16 | 43.08 ± 11.78 | 54.61 ± 6.79 | 51.68 ± 11.27 | 57.28 ± 4.57 | **61.02** ± 4.19 |
| Rating DVD | 3 | 4 | 32.22 ± 08.72 | 45.94 ± 7.48 | 41.23 ± 10.98 | 47.73 ± 6.20 | **49.76** ± 9.80 |
| | | 8 | 36.35 ± 12.50 | 46.23 ± 6.03 | 45.24 ± 9.76 | 47.11 ± 4.00 | **53.28** ± 4.66 |
| | | 16 | 42.79 ± 10.18 | 49.23 ± 6.68 | 45.19 ± 11.56 | 48.39 ± 3.74 | **53.52** ± 4.77 |
| Rating Electronics | 3 | 4 | 39.27 ± 10.15 | 39.89 ± 5.83 | 41.20 ± 10.69 | 37.40 ± 3.72 | **51.71** ± 7.20 |
| | | 8 | 28.74 ± 08.22 | 46.53 ± 5.44 | 45.41 ± 09.49 | 43.64 ± 7.31 | **54.78** ± 6.48 |
| | | 16 | 45.48 ± 06.13 | 48.71 ± 6.16 | 47.29 ± 10.55 | 44.83 ± 5.96 | **58.69** ± 2.41 |
| Rating Kitchen | 3 | 4 | 34.76 ± 11.20 | 40.41 ± 5.33 | 36.77 ± 10.62 | 44.72 ± 9.13 | **50.21** ± 09.63 |
| | | 8 | 34.49 ± 08.72 | 48.35 ± 7.87 | 47.98 ± 09.73 | 46.03 ± 8.57 | **53.72** ± 10.31 |
| | | 16 | 47.94 ± 08.28 | 52.94 ± 7.14 | 53.79 ± 09.47 | 49.85 ± 9.31 | **57.00** ± 08.69 |
| Overall Average | | 4 | 38.13 | 40.13 | 40.10 | 36.29 | **45.99** |
| | | 8 | 36.99 | 45.89 | 44.25 | 39.15 | **50.86** |
| | | 16 | 48.55 | 49.93 | 49.07 | 39.85 | **55.50** |

Table 1: Few-shot generalization performance across tasks not seen during training. $k$ is the number of examples per label for fine-tuning and $N$ is the number of classes for the task. On average, LEOPARD is significantly better than other models for few-shot transfer to new tasks.

| | $k$ | $\text{BERT}_{\text{base}}$ | $\text{MT-BERT}_{\text{softmax}}$ | MT-BERT | $\text{MT-BERT}_{\text{reuse}}$ | Proto-BERT | LEOPARD |
|---|---|---|---|---|---|---|---|
| **Natural Language Inference** | | | | | | | |
| Scitail | 4 | 58.53 ± 09.74 | 74.35 ± 5.86 | 63.97 ± 14.36 | **76.65 ± 2.45** | 76.27 ± 4.26 | 69.50 ± 9.56 |
| | 8 | 57.93 ± 10.70 | **79.11 ± 3.11** | 68.24 ± 10.33 | 76.86 ± 2.09 | 78.27 ± 0.98 | 75.00 ± 2.42 |
| | 16 | 65.66 ± 06.82 | **79.60 ± 2.31** | 75.35 ± 04.80 | 79.53 ± 2.17 | 78.59 ± 0.48 | 77.03 ± 1.82 |
| **Amazon Review Sentiment Classification** | | | | | | | |
| Books | 4 | 54.81 ± 3.75 | 68.69 ± 5.21 | 64.93 ± 8.65 | 74.79 ± 6.91 | 73.15 ± 5.85 | **82.54 ± 1.33** |
| | 8 | 53.54 ± 5.17 | 74.86 ± 2.17 | 67.38 ± 9.78 | 78.21 ± 3.49 | 75.46 ± 6.87 | **83.03 ± 1.28** |
| | 16 | 65.56 ± 4.12 | 74.88 ± 4.34 | 69.65 ± 8.94 | 78.87 ± 3.32 | 77.26 ± 3.27 | **83.33 ± 0.79** |
| Kitchen | 4 | 56.93 ± 7.10 | 63.07 ± 7.80 | 60.53 ± 9.25 | 75.40 ± 6.27 | 62.71 ± 9.53 | **78.35 ± 18.36** |
| | 8 | 57.13 ± 6.60 | 68.38 ± 4.47 | 69.66 ± 8.05 | 75.13 ± 7.22 | 70.19 ± 6.42 | **84.88 ± 01.12** |
| | 16 | 68.88 ± 3.39 | 75.17 ± 4.57 | 77.37 ± 6.74 | 80.88 ± 1.60 | 71.83 ± 5.94 | **85.27 ± 01.31** |

Table 2: Domain transfer evaluation (accuracy) on NLI and Sentiment classification datasets.

We use the following datasets (more details in Appendix): (1) entity typing: CoNLL-2003 (Sang and De Meulder, 2003), MIT-Restaurant (Liu et al., 2013); (2) rating classification: we use the review ratings for each domain from the Amazon Reviews dataset (Blitzer et al., 2007) and consider a 3-way classification based on the ratings; (3) text classification: social-media datasets from crowdflower[4].

Table 1 shows the performance. We can see that, on average, LEOPARD outperforms all the baselines, yielding significant improvements in accuracy. This shows LEOPARD's robustness to varying number of labels across tasks and across different text domains. Note that LEOPARD uses the same training tasks as MT-BERT but can adapt to new tasks with fewer examples, and improvements are highest with only 4 examples. Performance of prototypical networks is worse than most other fine-tuning methods on new training tasks. We hypothesize that this is because prototypical networks do not generate good class prototypes for new tasks and adaptation of class prototypes is important for improving performance. We also see that improved feature learning in MT-BERT with additional training tasks serves as a better initialization point for held-out tasks than BERT, and only tuning the softmax layer of this model is slightly better than tuning all parameters. Interestingly, on some tasks like Disaster classification, we observe BERT to perform better than other models, indicating negative transfer from the training tasks.

#### 4.3.2 Few-Shot Domain Transfer

We now evaluate performance on new domains of tasks seen at training time. For this, we consider two tasks of Sentiment Classification and NLI. For sentiment classification we use 4 domains of Amazon reviews (Blitzer et al., 2007) and for NLI we use a scientific entailment dataset (SciTail) (Khot et al., 2018). We introduce another relevant baseline here, $\text{MT-BERT}_{\text{reuse}}$, which reuses the trained softmax parameters of a related train task. Results are summarized in Table 2, we show two domains of sentiment classification and more results are in Appendix B. Note that the related train task, SST, only contains phrase-level sentiments and the models weren't trained to predict sentence-level sentiment, while the target tasks require sentence-level sentiment. We observe that LEOPARD performs better than the baselines on all domains of sentiment classification, while on Scitail MT-BERT models perform better, potentially because training consisted of many related NLI datasets. Note that prototypical networks is a competitive baseline here and its performance is better for these tasks in comparison to those in Table 1 as it has learned to generate prototypes for a similar task during training.

### 4.4 Ablation Study

For ablations we use the dev-set of 3 tasks: CoNLL-2003 entity typing, Amazon reviews DVD domain sentiment classification and SciTail NLI.

**Importance of softmax parameters**: Since the softmax generation is an important component of LEOPARD, we study how it affects performance. We remove the softmax generator and instead add a softmax weight and bias with zero initialization for each task. The model is trained in a similar way as LEOPARD. This method, termed LEOPARD-ZERO, is a naive application of MAML to this problem. Table 3 shows that this performs worse on new tasks, highlighting the importance of softmax generator.

[4]https://www.figure-eight.com/data-for-everyone/

| $k$ | Model | Entity Typing | Sentiment Classification | NLI |
|---|---|---|---|---|
| 16 | LEOPARD $_{10}$ | 37.62 $\pm$ 7.37 | 58.10 $\pm$ 5.40 | 78.53 $\pm$ 1.55 |
| | LEOPARD $_{5}$ | 62.49 $\pm$ 4.23 | 71.50 $\pm$ 5.93 | 73.27 $\pm$ 2.63 |
| | LEOPARD | 69.00 $\pm$ 4.76 | 76.65 $\pm$ 2.47 | 76.10 $\pm$ 2.21 |
| | LEOPARD-ZERO | 44.79 $\pm$ 9.34 | 74.45 $\pm$ 3.34 | 74.36 $\pm$ 6.67 |

Table 3: Ablations: LEOPARD$_\nu$ does not adapt layers $0-\nu$ (inclusive) in the inner loop (and fine-tuning), while LEOPARD adapts all parameters. Note that the outer loop still optimizes all parameters. For new tasks (like entity typing) adapting all parameters is better while for tasks seen at training time (like NLI) adapting fewer parameters is better. LEOPARD-ZERO is a model trained without the softmax-generator and a zero initialized softmax classifier, which shows the importance of softmax generator in LEOPARD.

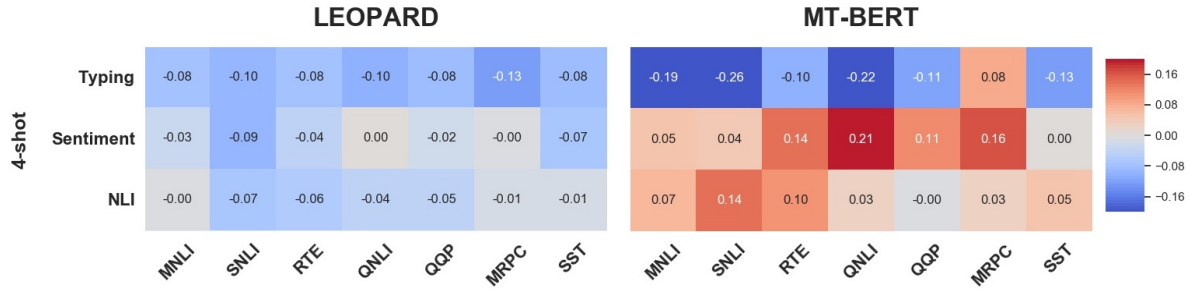


Figure 2: Analyzing target task performance as a function of training tasks (best viewed in color). Each column represents one held-out training task (name on $x$-axis) and each row corresponds to one target task (name on $y$-axis). Each cell is the relative change in performance on the target task when the corresponding training task is held-out, compared to training on all the train tasks. Dark blue indicates large drop, dark red indicates large increase and grey indicates close to no change in performance. In general, LEOPARD's performance is more consistent compared to MT-BERT indicating that meta-training learns more generalized initial parameters compared to multi-task training.

**Parameter efficiency**: We consider three variants of LEOPARD with parameter efficient training discussed in Sec 3.3. Denote LEOPARD$_\nu$ as the model which does not adapt layers 0 to $\nu$ (including word embeddings) in the inner loop of meta-training. Note that even for $\nu \neq 0$, the parameters are still optimized in the outer loop. Table 3 shows the results. Interestingly, for all tasks (except NLI) we find that adapting all parameters is better. This is potentially because the per-layer learning rate in LEOPARD also adjust the adaptation rates for each layer. On SciTail (NLI) we observe the opposite behaviour, suggesting that adapting fewer parameters is better for small $k$, potentially because training consisted of multiple NLI datasets.

**Importance of training tasks**: We study how target-task performance of MT-BERT and LEOPARD is dependent on tasks used for training. For this experiment, we held out each training task one by one and trained both models. The trained models are then evaluated for their performance on the target tasks (using the development set), following the same protocol as before. Fig. 2 shows a visualization of the relative change in performance when each training task is held out. We see that LEOPARD's performance is more consistent with respect to variation in training tasks, owing to the meta-training procedure that finds an initial point that performs equally well across tasks. Removing a task often leads to decrease in performance for LEOPARD as it decreases the number of meta-training tasks and leads to over-fitting to the training task-distribution. In contrast, MT-BERT's performance on target tasks varies greatly depending on the held-in training tasks.

## 5 Related Work

Meta-Learning approaches can be broadly classified as: optimization-based (Finn et al., 2017; Al-Shedivat et al., 2018; Nichol and Schulman, 2018; Rusu et al., 2019), model-based (Santoro et al., 2016; Ravi and Larochelle, 2017; Munkhdalai and Yu, 2017), and metric-learning based (Vinyals et al.,

2016; Snell et al., 2017; Sung et al., 2018). We refer to Finn (2018) for an exhaustive review. Recently, it has been shown that learning task-dependent model parameters improves few-shot learning (Rusu et al., 2019; Zintgraf et al., 2019). While existing methods train and evaluate on simulated datasets with limited diversity, there is recent interest for more realistic meta-learning applications (Triantafillou et al., 2019) and our work significantly advances this by training and evaluating on diverse and real NLP tasks.

Meta-learning applications in NLP have yielded improvements on specific tasks. Gu et al. (2018) used MAML to simulate low resource machine translation, Chen et al. (2018) learn HyperLSTM (Ha et al., 2016) model in a multi-task setting across various sentiment classification domains, and other recent approaches (Guo et al., 2018; Yu et al., 2018; Han et al., 2018; Obamuyide and Vlachos, 2019; Geng et al., 2019; Mi et al., 2019; Bao et al., 2020) meta-train for a *specific* classification task, such as relation classification, and do not generalize beyond the training task. Dou et al. (2019) train on a subset of GLUE tasks to generalize to other GLUE tasks and their approach does not consider unseen tasks. Transfer learning is a closely related research area. Self-supervised pre-training has been shown to learn general-purpose model parameters that improve downstream performance with fine-tuning (Peters et al., 2018; Howard and Ruder, 2018; Devlin et al., 2018; Radford et al., 2019; Yang et al., 2019; Raffel et al., 2019). Fine-tuning, however, typically requires large training data (Yogatama et al., 2019). Multi-task learning with BERT has been shown to improve performance for many related tasks (Phang et al., 2018; Wang et al., 2018a; Liu et al., 2019). We refer the reader to Ruder (2019) for a more thorough discussion of transfer learning and multi-task learning.

## 6 Conclusions

Learning general linguistic intelligence has been a long-term goal of NLP. While humans, with all their prior knowledge, can quickly learn to solve new tasks with very few examples, machine-learned models still struggle to demonstrate such intelligence. To this end, we proposed LEOPARD, a meta-learning approach, and found that it learns more general-purpose parameters that better prime the model to solve completely new tasks with few examples. While we see improvements using meta-learning, performance with few examples still lags behind human-level performance. We consider bridging this gap as a lucrative goal to demonstrate general linguistic intelligence, and meta-learning as a strong contender to achieve this goal.

## 7 Acknowledgements

We will like to thank Kalpesh Krishna, Tu Vu and Tsendsuren Munkhdalai for feedback on earlier drafts of this manuscript. This work was supported in part by the Chan Zuckerberg Initiative, and in part by the National Science Foundation under Grant No. IIS-1514053 and IIS-1763618. Any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect those of the sponsor.

## References

Maruan Al-Shedivat, Trapit Bansal, Yuri Burda, Ilya Sutskever, Igor Mordatch, and Pieter Abbeel. 2018. Continuous adaptation via meta-learning in nonstationary and competitive environments. In *Proceedings of the International Conference on Learning Representations*.

Antreas Antoniou, Harrison Edwards, and Amos Storkey. 2018. How to train your maml. *arXiv preprint arXiv:1810.09502*.

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. 2016. Layer normalization. *arXiv preprint arXiv:1607.06450*.

Yujia Bao, Menghua Wu, Shiyu Chang, and Regina Barzilay. 2020. Few-shot text classification with distributional signatures. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net.

Samy Bengio, Yoshua Bengio, Jocelyn Cloutier, and Jan Gecsei. 1992. On the optimization of a synaptic learning rule. In *Preprints Conf. Optimality in Artificial and Biological Neural Networks*, pages 6–8. Univ. of Texas.

John Blitzer, Mark Dredze, and Fernando Pereira. 2007. Biographies, bollywood, boom-boxes and blenders: Domain adaptation for sentiment classification. In *Proceedings of the 45th annual meeting of the association of computational linguistics*, pages 440–447.

Samuel R Bowman, Gabor Angeli, Christopher Potts, and Christopher D Manning. 2015. A large annotated corpus for learning natural language inference. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 632–642.

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

Junkun Chen, Xipeng Qiu, Pengfei Liu, and Xuanjing Huang. 2018. Meta multi-task learning for sequence modeling. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

Ido Dagan, Oren Glickman, and Bernardo Magnini. 2005. The pascal recognising textual entailment challenge. In *Machine Learning Challenges Workshop*, pages 177–190. Springer.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

William B Dolan and Chris Brockett. 2005. Automatically constructing a corpus of sentential paraphrases. In *Proceedings of the Third International Workshop on Paraphrasing (IWP2005)*.

Zi-Yi Dou, Keyi Yu, and Antonios Anastasopoulos. 2019. Investigating meta-learning algorithms for low-resource natural language understanding tasks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 1192–1197.

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

Chelsea Finn. 2018. *Learning to Learn with Gradients*. Ph.D. thesis, UC Berkeley.

Ruiying Geng, Binhua Li, Yongbin Li, Xiaodan Zhu, Ping Jian, and Jian Sun. 2019. Induction networks for few-shot text classification.

Danilo Giampiccolo, Bernardo Magnini, Ido Dagan, and Bill Dolan. 2007. The third pascal recognizing textual entailment challenge. In *Proceedings of the ACL-PASCAL workshop on textual entailment and paraphrasing*, pages 1–9.

Danilo Giampiccolo, Hoa Trang Dang, Bernardo Magnini, Ido Dagan, Elena Cabrio, and Bill Dolan. 2008. The fourth pascal recognizing textual entailment challenge. In *TAC*. Citeseer.

Jiatao Gu, Yong Wang, Yun Chen, Kyunghyun Cho, and Victor OK Li. 2018. Meta-learning for low-resource neural machine translation. *arXiv preprint arXiv:1808.08437*.

Jiang Guo, Darsh Shah, and Regina Barzilay. 2018. Multi-source domain adaptation with mixture of experts. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4694–4703.

David Ha, Andrew Dai, and Quoc V Le. 2016. Hypernetworks. *arXiv preprint arXiv:1609.09106*.

R Bar Haim, Ido Dagan, Bill Dolan, Lisa Ferro, Danilo Giampiccolo, Bernardo Magnini, and Idan Szpektor. 2006. The second pascal recognising textual entailment challenge. In *Proceedings of the Second PASCAL Challenges Workshop on Recognising Textual Entailment*.

Xu Han, Hao Zhu, Pengfei Yu, Ziyun Wang, Yuan Yao, Zhiyuan Liu, and Maosong Sun. 2018. Fewrel: A large-scale supervised few-shot relation classification dataset with state-of-the-art evaluation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4803–4809.

Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin De Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. 2019. Parameter-efficient transfer learning for nlp. In *International Conference on Machine Learning*.

Jeremy Howard and Sebastian Ruder. 2018. Universal language model fine-tuning for text classification. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 328–339.

Katharina Kann, Kyunghyun Cho, and Samuel R Bowman. 2019. Towards realistic practices in low-resource natural language processing: The development set. *arXiv preprint arXiv:1909.01522*.

Tushar Khot, Ashish Sabharwal, and Peter Clark. 2018. Scitail: A textual entailment dataset from science question answering. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

Zhenguo Li, Fengwei Zhou, Fei Chen, and Hang Li. 2017. Meta-sgd: Learning to learn quickly for few-shot learning. *arXiv preprint arXiv:1707.09835*.

Jingjing Liu, Panupong Pasupat, Scott Cyphers, and Jim Glass. 2013. Asgard: A portable architecture for multilingual dialogue systems. In *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 8386–8390. IEEE.

Xiaodong Liu, Pengcheng He, Weizhu Chen, and Jianfeng Gao. 2019. Multi-task deep neural networks for natural language understanding. *arXiv preprint arXiv:1901.11504*.

Fei Mi, Minlie Huang, Jiyong Zhang, and Boi Faltings. 2019. Meta-learning for low-resource natural language generation in task-oriented dialogue systems. *arXiv preprint arXiv:1905.05644*.

Tsendsuren Munkhdalai and Hong Yu. 2017. Meta networks. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 2554–2563. JMLR. org.

Alex Nichol and John Schulman. 2018. Reptile: a scalable metalearning algorithm. *arXiv preprint arXiv:1803.02999*, 2.

Abiola Obamuyide and Andreas Vlachos. 2019. Model-agnostic meta-learning for relation classification with limited supervision. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5873–5879.

Matthew E Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of NAACL-HLT*, pages 2227–2237.

Jason Phang, Thibault Févry, and Samuel R Bowman. 2018. Sentence encoders on stilts: Supplementary training on intermediate labeled-data tasks. *arXiv preprint arXiv:1811.01088*.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners. *OpenAI Blog*, 1(8).

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. 2019. Exploring the limits of transfer learning with a unified text-to-text transformer. *arXiv preprint arXiv:1910.10683*.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. Squad: 100,000+ questions for machine comprehension of text. *arXiv preprint arXiv:1606.05250*.

Sachin Ravi and Hugo Larochelle. 2017. Optimization as a model for few-shot learning. In *Proceedings of the International Conference on Learning Representations*.

Sebastian Ruder. 2019. *Neural Transfer Learning for Natural Language Processing*. Ph.D. thesis, NATIONAL UNIVERSITY OF IRELAND, GALWAY.

Andrei A. Rusu, Dushyant Rao, Jakub Sygnowski, Oriol Vinyals, Razvan Pascanu, Simon Osindero, and Raia Hadsell. 2019. Meta-learning with latent embedding optimization. In *International Conference on Learning Representations*.

Erik F Tjong Kim Sang and Fien De Meulder. 2003. Introduction to the conll-2003 shared task: Language-independent named entity recognition. In *Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003*, pages 142–147.

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

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

Jake Snell, Kevin Swersky, and Richard Zemel. 2017. Prototypical networks for few-shot learning. In *Advances in Neural Information Processing Systems*, pages 4077–4087.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 conference on empirical methods in natural language processing*, pages 1631–1642.

Flood Sung, Yongxin Yang, Li Zhang, Tao Xiang, Philip HS Torr, and Timothy M Hospedales. 2018. Learning to compare: Relation network for few-shot learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1199–1208.

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

Eleni Triantafillou, Tyler Zhu, Vincent Dumoulin, Pascal Lamblin, Utku Evci, Kelvin Xu, Ross Goroshin, Carles Gelada, Kevin Swersky, Pierre-Antoine Manzagol, and Hugo Larochelle. 2019. Meta-dataset: A dataset of datasets for learning to learn from few examples.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008.

Oriol Vinyals, Charles Blundell, Timothy Lillicrap, Daan Wierstra, et al. 2016. Matching networks for one shot learning. In *Advances in neural information processing systems*, pages 3630–3638.

Alex Wang, Jan Hula, Patrick Xia, Raghavendra Pappagari, R Thomas McCoy, Roma Patel, Najoung Kim, Ian Tenney, Yinghui Huang, Katherin Yu, et al. 2018a. Can you tell me how to get past sesame street? sentence-level pretraining beyond language modeling. *arXiv preprint arXiv:1812.10860*.

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

Alex Warstadt, Amanpreet Singh, and Samuel R Bowman. 2019. Neural network acceptability judgments. *Transactions of the Association for Computational Linguistics*, 7:625–641.

Adina Williams, Nikita Nangia, and Samuel R Bowman. 2017. A broad-coverage challenge corpus for sentence understanding through inference. *arXiv preprint arXiv:1704.05426*.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Ruslan Salakhutdinov, and Quoc V Le. 2019. Xlnet: Generalized autoregressive pretraining for language understanding. *arXiv preprint arXiv:1906.08237*.

Dani Yogatama, Cyprien de Masson d'Autume, Jerome Connor, Tomás Kociský, Mike Chrzanowski, Lingpeng Kong, Angeliki Lazaridou, Wang Ling, Lei Yu, Chris Dyer, and Phil Blunsom. 2019. Learning and evaluating general linguistic intelligence. *CoRR*, abs/1901.11373.

Mo Yu, Xiaoxiao Guo, Jinfeng Yi, Shiyu Chang, Saloni Potdar, Yu Cheng, Gerald Tesauro, Haoyu Wang, and Bowen Zhou. 2018. Diverse few-shot text classification with multiple metrics. *arXiv preprint arXiv:1805.07513*.

Luisa M Zintgraf, Kyriacos Shiarlis, Vitaly Kurin, Katja Hofmann, and Shimon Whiteson. 2019. Cavia: Fast context adaptation via meta-learning. In *International Conference on Machine Learning*.

# A Appendix

## A Datasets

**Data Augmentation**: Meta-learning benefits from training across many tasks. We thus create multiple versions of tasks with more than 2 classes by considering classifying between every pair of labels as a task. Existing methods (Vinyals et al., 2016; Snell et al., 2017; Finn et al., 2017) treat each random sample of labels from a pool of labels (for example in image classification) as a task. In order to create more diversity during training, we also create multiple versions of each dataset that has more than 2 classes, by considering classifying between every possible pair of labels as a training task. This increases the number of tasks and allows for more per-label examples in a batch during training. In addition, since one of the goals is to learn to classify phrases in a sentence, we modify the sentiment classification task (SST-2) in GLUE, which contains annotations of sentiment for phrases, by providing a sentence in which the phrase occurs as part of the input. That is, the input is the sentence followed by a separator token

| Input | Label |
|---|---|
| are there any [authentic mexican][1] restaurants in [the area][2] | [1]Cuisine, [2]Location |
| are there any authentic mexican restaurants in the area [SEP] authentic mexican | Cuisine |
| are there any authentic mexican restaurants in the area [SEP] the area | Location |

Table 4: An example of an input from the MIT restaurants dataset. The first line is the actual example with two mentions. The next two lines are the input to the models – one for each mention.

(Devlin et al., 2018) followed by the phrase to classify. An example of the input to all the models for the entity typing tasks can be found in Table 4

We use the standard train, dev data split for GLUE and SNLI (Wang et al., 2018b; Bowman et al., 2015). For our ablation studies, on our target task we take 20% of the training data as validation for early stopping and sample from the remaining 80% to create the few-shot data for fine-tuning. For training MT-BERT we use dev data of the training task as the validation set. For meta-learning methods, prototypical network and LEOPARD, we use additional validation datasets as is typical in meta learning (Finn et al., 2017; Snell et al., 2017). We use unlabelled Amazon review data from apparel, health, software, toys, video as categorization tasks and labelled data from music, toys, video as sentiment classification task.

Details of the datasets are present in Table 5.

| Dataset | Labels | Training Size | Validation Size | Testing Size | Source |
|---|---|---|---|---|---|
| ARSC Domains | 2 | 800 | 200 | 1000 | (Blitzer et al., 2007) |
| CoLA | 2 | 8551 | 1042 | — | (Warstadt et al., 2019) |
| MRPC | 2 | 3669 | 409 | — | (Dolan and Brockett, 2005) |
| QNLI | 2 | 104744 | 5464 | — | (Rajpurkar et al., 2016; Wang et al., 2018b) |
| QQP | 2 | 363847 | 40431 | — | (Wang et al., 2018b) |
| RTE | 2 | 2491 | 278 | — | (Dagan et al., 2005; Haim et al., 2006; Giampiccolo et al., 2007; Giampiccolo et al., 2008) |
| SNLI | 3 | 549368 | 9843 | — | (Bowman et al., 2015) |
| SST-2 | 2 | 67350 | 873 | — | (Socher et al., 2013) |
| MNLI (m/mm) | 3 | 392703 | 19649 | — | (Williams et al., 2017) |
| Scitail | 2 | 23,596 | 1,304 | 2,126 | (Khot et al., 2018) |
| Airline | 3 | 7320 | — | 7320 | https://www.figure-eight.com/data-for-everyone/ |
| Disaster | 2 | 4887 | — | 4887 | https://www.figure-eight.com/data-for-everyone/ |
| Political Bias | 2 | 2500 | — | 2500 | https://www.figure-eight.com/data-for-everyone/ |
| Political Audience | 2 | 2500 | — | 2500 | https://www.figure-eight.com/data-for-everyone/ |
| Political Message | 9 | 2500 | — | 2500 | https://www.figure-eight.com/data-for-everyone/ |
| Emotion | 13 | 20000 | — | 20000 | https://www.figure-eight.com/data-for-everyone/ |
| CoNLL | 4 | 23499 | 5942 | 5648 | (Sang and De Meulder, 2003) |
| MIT-Restaurant | 8 | 12474 | — | 2591 | (Liu et al., 2013) https://groups.csail.mit.edu/sls/downloads/restaurant/ |

Table 5: Dataset statistics for all the datasets used in our analysis. "-" represent data that is either not available or not used in this study. We have balanced severely unbalanced datasets(Political Bias and Audience) as our training data is balanced. To create training data for few shot experiments we sample 10 datasets for each k-shot. *Sec A for more details

### A.1 Test Datasets

The tasks and datasets we used for evaluating performance on few-shot learning are as follows:

1. Entity Typing: We use the following datasets for entity typing: CoNLL-2003 (Sang and De Meulder, 2003) and MIT-Restaurant (Liu et al., 2013). Note that we consider each mention as a separate labelled example. CoNLL dataset consists of text from news articles while MIT dataset contains text from restaurant queries.

2. Sentiment Classification: We use the sentiment annotated data from Amazon Reviews dataset (Blitzer et al., 2007) which contains user reviews and the binary sentiment for various domains of products. We use the Books, DVD, Electronics, and Kitchen & Housewares domains, which are commonly used domains in the literature (Yu et al., 2018).

3. Rating Classification: We use the ratings from the Amazon Reviews dataset (Blitzer et al., 2007) which is not annotated with overall sentiment, and consider classifying into 3 classes: rating $\leq 2$, rating $= 4$ and rating $= 5$.

4. Text Classification: We use multiple text classification datasets from crowdflower[5]. These involve classifying sentiments of tweets towards an airline, classifying whether a tweet refers to a disaster event, classifying emotional content of text, classifying the audience/bias/message of social media messages from politicians. These tasks are quite different from the training tasks both in terms of the labels as well as the input domain.

5. NLI: We use the SciTail dataset (Khot et al., 2018), which is a dataset for entailment created from science questions.

## B Additional Results

| Amazon Review Sentiment Classification | | | | | | | |
|---|---|---|---|---|---|---|---|
| | | $\text{BERT}_{\text{base}}$ | $\text{MT-BERT}_{\text{softmax}}$ | MT-BERT | $\text{MT-BERT}_{\text{resue}}$ | Proto-BERT | LEOPARD |
| Books | 4 | 54.81 $\pm$ 3.75 | 68.69 $\pm$ 5.21 | 64.93 $\pm$ 8.65 | 74.79 $\pm$ 6.91 | 73.15 $\pm$ 5.85 | 82.54 $\pm$ 1.33 |
| | 8 | 53.54 $\pm$ 5.17 | 74.86 $\pm$ 2.17 | 67.38 $\pm$ 9.78 | 78.21 $\pm$ 3.49 | 75.46 $\pm$ 6.87 | 83.03 $\pm$ 1.28 |
| | 16 | 65.56 $\pm$ 4.12 | 74.88 $\pm$ 4.34 | 69.65 $\pm$ 8.94 | 78.87 $\pm$ 3.32 | 77.26 $\pm$ 3.27 | 83.33 $\pm$ 0.79 |
| DVD | 4 | 54.98 $\pm$ 3.96 | 63.68 $\pm$ 5.03 | 66.36 $\pm$ 7.46 | 71.74 $\pm$ 8.54 | 74.38 $\pm$ 2.44 | 80.32 $\pm$ 1.02 |
| | 8 | 55.63 $\pm$ 4.34 | 67.54 $\pm$ 4.06 | 68.37 $\pm$ 6.51 | 75.36 $\pm$ 4.86 | 75.19 $\pm$ 2.56 | 80.85 $\pm$ 1.23 |
| | 16 | 58.69 $\pm$ 6.08 | 70.21 $\pm$ 1.94 | 70.29 $\pm$ 7.40 | 76.20 $\pm$ 2.90 | 75.26 $\pm$ 1.07 | 81.25 $\pm$ 1.41 |
| Electronics | 4 | 58.77 $\pm$ 6.10 | 61.63 $\pm$ 7.30 | 64.13 $\pm$ 10.34 | 72.82 $\pm$ 6.34 | 65.68 $\pm$ 6.80 | 74.88 $\pm$ 16.59 |
| | 8 | 59.00 $\pm$ 5.78 | 66.29 $\pm$ 5.36 | 64.21 $\pm$ 10.49 | 75.07 $\pm$ 3.40 | 68.54 $\pm$ 5.61 | 81.29 $\pm$ 1.65 |
| | 16 | 67.32 $\pm$ 4.18 | 69.61 $\pm$ 3.54 | 71.12 $\pm$ 7.29 | 75.40 $\pm$ 2.43 | 67.84 $\pm$ 7.23 | 81.86 $\pm$ 1.56 |
| Kitchen | 4 | 56.93 $\pm$ 7.10 | 63.07 $\pm$ 7.80 | 60.53 $\pm$ 9.25 | 75.40 $\pm$ 6.27 | 62.71 $\pm$ 9.53 | 78.35 $\pm$ 18.36 |
| | 8 | 57.13 $\pm$ 6.60 | 68.38 $\pm$ 4.47 | 69.66 $\pm$ 8.05 | 75.13 $\pm$ 7.22 | 70.19 $\pm$ 6.42 | 84.88 $\pm$ 1.12 |
| | 16 | 68.88 $\pm$ 3.39 | 75.17 $\pm$ 4.57 | 77.37 $\pm$ 6.74 | 80.88 $\pm$ 1.60 | 71.83 $\pm$ 5.94 | 85.27 $\pm$ 1.31 |

Table 6: Domain transfer evaluation (accuracy) on Sentiment classification datasets.

| | MNLI(m/mm) | QQP | QNLI | SST-2 | CoLA | MRPC | RTE | SNLI | Average |
|---|---|---|---|---|---|---|---|---|---|
| MT-BERT | 82.11 | 89.92 | 89.62 | 90.7 | 81.30 | 84.56 | 78.34 | 89.97 | **85.82** |

Table 7: Dev-set accuracy on the set of train tasks for multi-task BERT.

Table 6 shows the accuracy on all the four amazon sentiment classification tasks.
Table 7 shows the dev-set accuracy of our trained MT-BERT model on the various training tasks.
Figure 3 shows the target task performance as a function of training tasks for all $k$. Note that the effect of training tasks starts to decrease as $k$ increases.

## C Hyperparameters

Table 8 shows the hyper-parameter search range as well as the best hyper-parameters for MT-BERT, Proto-BERT and LEOPARD. We use same hyperparameters for prototypical networks except those not relevant to them. For fine-tuning we separately tune number of iterations, and batch size for each k shot for all the baselines. We also tuned warm-up (Devlin et al., 2018) in $\{0, 0.1\}$ and used 0.1 for all the methods. For MT-BERT we found 10 epochs , batch size 8 to be best for 4-shot, 5 epochs, batch size 8 to be best for 8-shot and 5 epoch with 16 batch size gave the best performance for 16 shot. For $\text{MT-BERT}_{\text{softmax}}$ we found 125 epoch, batch size 4 to be best for 4-shot, 125 epochs, batch size 4 to be best for 8-shot and 125 epochs with batch size 4 gave the best performance for 16-shot. For $\text{BERT}_{\text{base}}$ 10 epochs, batch size 8 for 4 shot, 5 epochs, 16 batch size for 8 shot and 10 epochs, batch size 16 for 16 shot gave the best performance. For $\text{MT-BERT}_{\text{reuse}}$ we found 10 epochs , batch size 8 to be best for 4-shot, 5 epochs, batch size 8 to be best for 8-shot and 5 epoch with 16 batch size gave the best performance for 16 shot. Note, for LEOPARD we use learned per-layer learning rates with SGD. We use the following values: 150 epochs for 4-shot, 100 epochs for 8-shot, 100 epochs for 16-shot.

[5]https://www.figure-eight.com/data-for-everyone/

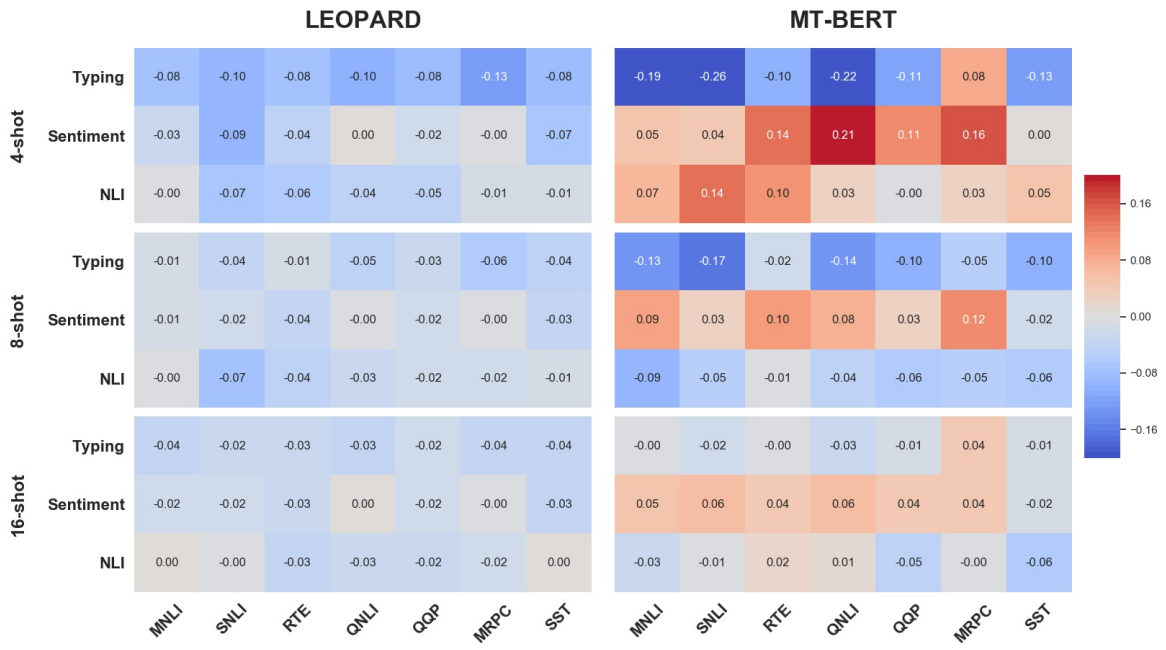


Figure 3: Analyzing target task performance as a function of training tasks (best viewed in color). Heatmaps on the left are for LEOPARD and on the right are for MT-BERT. Each column represents one held-out training task (name on $x$-axis) and each row corresponds to one target task (name on $y$-axis). Each cell is the relative change in performance on the target task when the corresponding training task is held-out, compared to training on all the train tasks. Dark blue indicates large drop, dark red indicates large increase and grey indicates close to no change in performance. In general, LEOPARD's performance is more consistent compared to MT-BERT indicating that meta-training learns more generalized initial parameters compared to multi-task training.

| Parameter | Search Space | MT-BERT | Proto-BERT | LEOPARD |
|---|---|---|---|---|
| Attention dropout | [0.1, 0.2, 0.3] | 0.2 | 0.3 | 0.1 |
| Batch Size | [16, 32] | 32 | 16 | 10 |
| Class Embedding Size | [128, 256] | — | 256 | 256 |
| Hidden Layer Dropout | [0.1, 0.2, 0.3] | 0.1 | 0.2 | 0.1 |
| Inner Loop Learning Rate | — | — | — | Meta-SGD (per-layer) |
| Min Adapted Layer ($\nu$) | [0, 5, 8, 10, 11] | — | — | 0 |
| Outer Loop Learning Rate | [1e-4, 1e-5, 2e-5, 4e-5, 5e-5] | 2e-05 | 2e-05 | 1e-05 |
| Adaptation Steps ($G$) | [1, 4, 7] | — | — | 7 |
| Top layer [CLS] dropout | [0.45, 0.4, 0.3, 0.2, 0.1] | 0.1 | 0.2 | 0.1 |
| Train Word Embeddings(Inner Loop) | [True, False] | — | — | True |
| Data Sampling | [Square Root, Uniform] | Square Root | Square Root | Square Root |
| Lowercase text | False | False | False | False |

Table 8: Hyper-parameter search space and best hyper-parameters for all models.