Exploring First Order Gradient Approximation Meta Learning for Robust QA Systems

Stanford CS224N Default Project

Christian Gabor
Department of Computer Science
Stanford University
`gaborc@stanford.edu`

Abstract

Finetuning pretrained language models such as BERT have shown good performance on new target domains. However, one limitation of finetuning has been the requirement of sufficient downstream task data for good generalization. This project explores meta learning for fast adaption on a pretrained language model DistilBERT with limited training examples. This project explores a meta learning approach called REPTILE as a first order approximation of the gradient based Model Agnostitic Meta Learning (MAML) approach. Current findings suggest Reptile may not improve test accuracy without sufficient hyper-parameter tuning. A new method introduced RandEPTILE adds random start points for Reptile which shows improvement on the validation accuracy.

1 Key Information to include

- This project was explored individually for Stanford CS 224n Winter 2021

2 Introduction

A common technique for training downstream NLP tasks involves adapting weights for a new task domain starting with a pretrained language model such as BERT. However, this approach can be challenging when the downstream NLP task does not have sufficient training data. The simple approach of retraining a model over new samples can require a large set of labeled data to reach good inference accuracy [1] and it could be prohibitively expensive to collect labeled samples. Another issue found after fine tuning is that the model may not generalize to new input domains for the same downstream task. This paper explores the problem of training question answering systems as a downstream task on new input domains where there are few labeled training examples.

Recent work with NLP transformer models have shown promise for meta learning on few shot downstream tasks. Meta learning in this setting involves searching for model parameters that enable efficient training with few samples. A well known gradient base approach Model Agnostic Meta Learning (MAML) [2] searches for good initial starting points of model parameters that can adapt to tasks. This technique involves running gradient steps on training data starting from initial model parameters, then using validation data for backpropagation over a new starting point. A challenge with this approach is that it since it involves backpropagation through multiple steps of gradient descent, MAML can be memory intensive and expensive to perform on large transformer models.

Other work on gradient based meta learning in NLP settings have used first order approximations to the gradient over inner gradient descent steps to improve performance. Techniques include LEOPARD [3] and REPTILE [4]. These studies have shown first order gradient approximations achieve equivalent performance to MAML on few shot learning tasks with few training examples used during fine tuning a language model for classification tasks.

In this study I explore whether REPTILE can improve performance for fine tuning distilBERT model parameters for question answering with few training examples. I also introduce a new method called RandEPTILE which adds noise to the starting point of REPTILE to avoid over fitting on the training examples and also incorporates the validation accuracy during the meta learning step similar to MAML. REPTILE is also constrained to only one task for Question Answering and uses the entire DistilBERT question answering model as the staring parameters which adds additional challenges to meta learning.

3 Related Work

Model Agnostic Meta Learning is a gradient based update approach that optimizes a starting point for model parameters θ through backpropagation after k shots of task updates.

Learning to Few-Shot Learn Across Diverse Natural Language Classification Tasks [3] provides a meta learning approach called LEOPARD for k-shot learning, which attempts to find task dependent initialization point across task domains. This approach is similar to MAML, however uses first order gradient approximation to update parameter initialization. The meta learning model learns a shared parameter initialization generator to predict initialization points for each k shot learning task.

Like MAML, LEOPARD[3] involves an inner backpropagation update loop among a set of k shot training examples. A task dependent classification loss is computed over the samples and several iterations of gradient updates are applied to unfrozen layers in the transformer model. After this inner loop is complete a validation loss is computed on the model which provides the loss signal to the meta learning parameters. This meta learning model predicts an initialization point for the classification layer with weights W and b that minimize the expected validation loss on later K shot learning inner loops. The outer step attempts to find a good initialization starting point for each task so that the model can avoid over-fitting when only given a few training examples.

In LEOPARD the weight initialization over n tasks is generated using the following equation,

$$w_i^n, b_i^n = \frac{1}{C_i^n} \sum_{x_j \in C_i^n} MLP(f_\theta(x_j))$$

where C represents the task domain and MLP is a multi-layer perception. In this equation, a separate set of weight initializations are generated for each task domain.

REPTILE is an approach that approximates MAML through a simple update rule and has demonstrated similar results to MAML [5]. The REPTILE update rule can be formulated with the following equation.

$$\theta = \theta + \alpha \frac{1}{|\{T_i\}|} \sum_{T_i} (\theta_i^{(k)} - \theta)$$

In the above formula, $\theta_i^{(k)}$ represents the model weights after k steps of gradient descent over the starting model parameters θ. The Reptile algorithm involves computing inner update steps to acquire $\theta_i^{(k)}$ and the meta learning step updates the starting point using this equation.

Dou et al. explore the first order gradient approximation meta learning with BERT for meta learning and demonstrate these approaches achieve higher performance than vanilla baseline pretraining on limited downstream data [4]. Their approach involves REPTILE [5] and has shown superior accuracy in experimental results despite its simplicity.

This study focuses on the later REPTILE approach over LEOPARD and entire model parameters of the distilBERT pretrained model are updated during the meta learning step.

4 Approach

In this study, a pretrained distilBERT model is used to train on the downstream task of question answering. There are three larger datasets SQuAD, NewsQA and Natural Questions which each include 50,000 training samples to train a baseline model with fine tuning the pretrained transformer language model.

Using the fine tuned model as a baseline starting point for θ I use the following REPTILE update rule to search for improved initial points during fine tuning.

$$\theta = \theta + \alpha \sum_{T_i} (\theta^{(k)} - \theta)$$

This equation is simplified version of REPTILE that does not distinguish θ_i in the update step for each task in multi task learning. The hyper parameter α can be tuned since the number of training examples seen during the inner gradient steps is fixed in size.

An alternative approach explored in this study is alternating between the in domain datasets and out of domain datasets as separate tasks that share the same initial parameters. This can be formalized as the following equation.

$$\theta = \theta + \alpha \sum_{T_i} (\theta_i^{(k)} - \theta)$$

Where $i \in \{1, 2\}$ over two separate QA tasks. The initialization of θ could involve the language model before it has seen any QA tasks, or reusing the same θ initialization after the model has been trained fully on the in domain tasks.

A 2D visualization of this update rule is shown in figure 1.

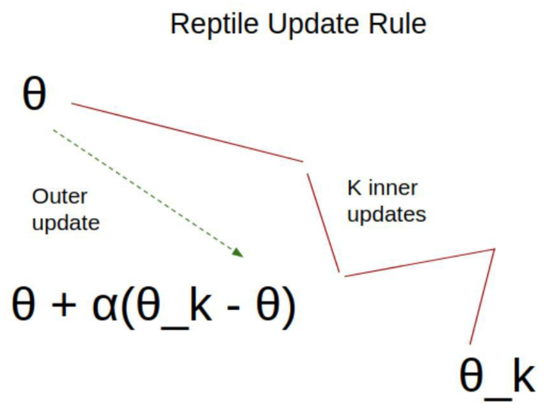

Figure 1: Reptile Update using one task in a 2D space

One limitation of this approach is that with only one task for training and because the number of samples is small, the outer update step may begin to converge to $\theta^{(k)}$. Because the validation dataset is not used in the REPTILE algorithm, the model could begin to over fit on the training data.

As an approach to avoid over fitting, I introduce a new method called RandEPTILE used to select an initial starting point before performing the k inner update steps. This new starting point is denoted θ^n. Finding an initial starting point can be done by generating a normal distribution added to the weights which perturbs the starting point of θ. The following starting point is used instead of the meta update used in REPTILE.

$$\theta^n = \theta + \beta \mathcal{N}(\mu, \sigma^2)$$

In this setting α is proportional to the learning rate of the inner update steps and $\mu = 0, \sigma^2 = 1$. The advantage of this approach is that it prevents the inner update steps from becoming stuck in local optima during training which can occur with only one task after the training loss converges to zero.

The RandEPTILE update rule is changed to the following form, where $\theta_i^{(s)}$ is the model parameters after k steps of gradient descent starting from initial point θ^n. This is the same form as REPTILE, but with a new starting point on each meta learning step.

$$\theta = \theta + \alpha \sum_{T_i} (\theta_i^{(s)} - \theta)$$

A 2D visualization of this new update rule called RandEPTILE is shown in figure 2.

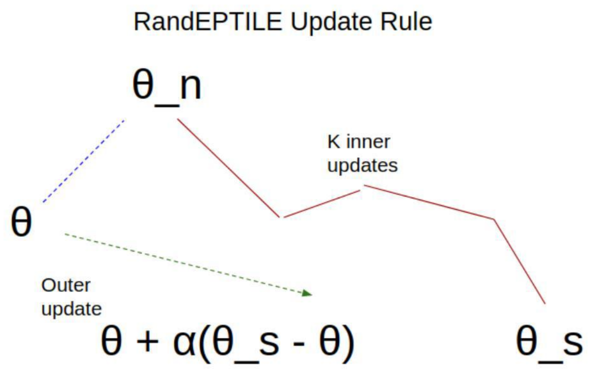

Figure 2: RandEPTILE Update using one task in a 2D space

One limitation of this approach is that the added noise to the model parameters may cause the performance to degrade over time. For this reason, the validation accuracy is used after k steps of SGD and the model is restored to a previous checkpoint if the validation accuracy falls too low. This re-incorporates the validation accuracy used in other meta learning approaches such as LEOPARD, however the model does not ever perform backpropagation over the validation dataset.

One observation is that the noise is added to all parameters of the model in this study. This may be a sub-optimal regime and it may be better to only add noise the final layers of the model.

5 Experiments

5.1 Data

For the RobustQA task I use the SQuAD, NewsQA and Natural Questions datasets which represent the in domain datasets. Each of these datasets contain 50,000 training examples for fine tuning however they are not used for the final downstream task. Instead the goal is to perform well on the DuoRC, RACE and RelationExtraction datasets which contain 127 training examples each.

For these experiments, a baseline model is pretrained first on the three datasets totaling 150,000 training examples. Using the weights of the baseline model as a starting point, I then perform various experiments on improving the validation accuracy of the remaining 3 smaller datasets which have a total of 382 validation samples combined. Finally for testing, the model with the best validation accuracy is run on test samples from DuoRC, RACE and RelationExtraction which have 1248, 419 and 2693 test samples that have not been seen during training or meta learning steps.

5.2 Evaluation method

Two metrics are using for evaluation of the model performance. I use the provided F1 and exact match (EM) scores to verify the model has improved in accuracy. When using the validation data as a starting point later in the model, I evaluate the F1 metric to save a checkpoint over the best weights instead of the EM score.

5.3 Experimental details

Various learning rates, batch sizes and other hyper parameters were evaluated during these experiments. Here I report the performance of the models using the best hyper parameters observed during evaluation.

I set the inner step number k to be 5 for all models and the batch size to 32. During the inner learning steps the model sees 160 total training examples. A batch size of 32 was the largest set that could fit into GPU memory for the gradients over the distilBERT model.

For REPTILE I used a learning rate of 8e-5 with stochastic gradient descent over mini batches with no momentum. I set the outer step size α equal to 0.25 that is fixed during each meta update step.

For RandEPTILE I set the learning rate to 3e-4 and β equal to 5e-4. These hyper parameters were chosen to have the random noise similar in magnitude to the learning rate of the inner update steps. This model also uses stochastic gradient descent over mini batches and α equal to 0.25 which were the same hyper parameters used in the REPTILE experiments. In addition, I introduce a margin in the F1 score to 1.0 so that if the model begins to degrade beyond this point, the previous model parameters are restored.

In searching for hyper parameters, I explored freezing layers of the model in REPTILE, however for the final results I chose to unfreeze all parameters of the transformer model so that θ includes the full pretrained model. This technique was not explored for adding randomness to the weights in RandREPTILE and could be a further study.

These models were also trained for at least 500 training steps before being stopped. The final models took between 10 and 20 hours to train on an NC6 Azure machine.

5.4 Results

Figure 3 and 4 show the F1 Score and EM score after inner updates between REPTILE and RandEPTILE. These update steps both began with the parameters of the baseline model from fine tuning on the in domain datasets.

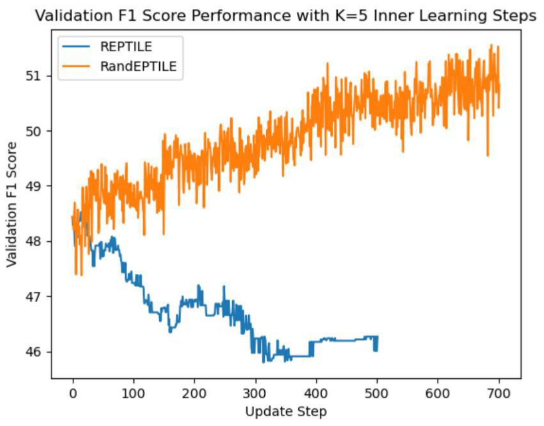

Figure 3: RandEPTILE Update using one task in a 2D space

It can be observed that the RandEPTILE improves the baseline over the basic REPTILE algorithm on the validation dataset. REPTILE appears to slowly degrade in performance and could be over fitting on the training data after many gradient steps.

Another experiment was done with RandEPTILE using only the pretrained distilBERT model. In this setting, the model randomly switched between two tasks of the in domain dataset and the out of domain dataset for the inner update steps. The results after 15 hours of training are shown in figure 5.

The following table shows the final performance of each approach.

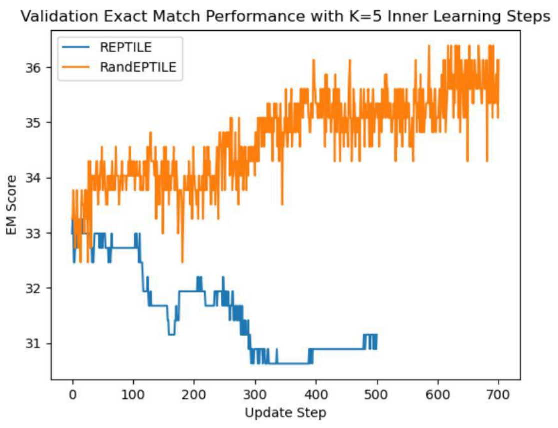

Figure 4: RandEPTILE Update using one task in a 2D space

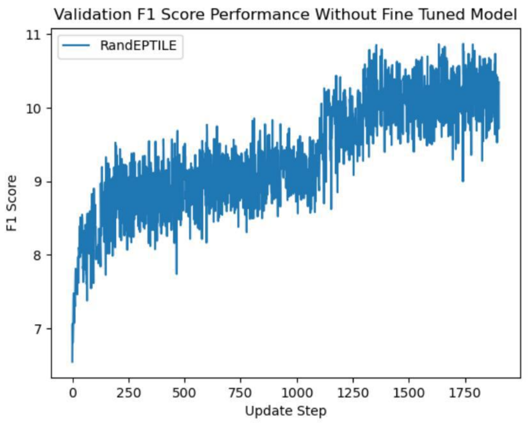

Figure 5: RandEPTILE Update using one task in a 2D space

Name	F1 Val	EM Val	F1 Test	EM Test
Baseline	48.43	33.25	**59.187**	40.28
Reptile	48.92	33.77	-	-
RandEPTILE	**51.55**	**36.39**	58.95	**40.99**

These results show that the validation accuracy of RandEPTILE improved however the F1 test score did worse than the baseline. There is a marginal increase in the exact match score over the baseline.

A possible observation is that the model did not perform enough update steps to improve over the baseline. Only 700 steps were used for the final RandEPTILE training procedure but the performance appears to still be improving at 700 steps. It is also possible that because I restore models using the validation accuracy from previous checkpoints that the validation accuracy is no longer an unbiased metric on how well it will perform on the test dataset despite infrequent reset points.

6 Analysis

Observing the outputs of the model shows that RandEPTILE is choosing answers that are frequently very long. For instance, the output of the model in the validation set chooses answers such as the following.

Ginger continues west with Harry. She claims she is 16, and is running away from home to a commune in Boulder.Harry and Ginger

1987 after the Herald of Free Enterprise disaster, when Townsend Thoresen was renamed PO European Ferries, until 1999

Because the model is being saved at points where it has a high F1 score match, the model could be attempting attempting to fit in more words into the answer without having a high confidence over the span of the true answer. These are usually incomplete sentences and are not accurate answers to a QA system even if the location of the match is approximately correct. This observation indicates that a better metric than F1 score to choose when to restore model weights might change the performance of the model.

7 Conclusion

The task of training a deep learning model using only several hundred training examples is a challenging task and remains an area worth more exploration. RandREPTILE was introduced to avoid potential over fitting encountered during REPTILE updates for first order gradient approximation in meta learning. We might expect that SGD adds enough noise to the inner batch updates, however after many epochs with only several hundred training examples, this noise may not be sufficient to keep the model from over fitting as seen during the basic REPTILE updates.

The results so far in these findings are inconclusive whether this REPTILE or RandREPTILE improved the performance on the model during test time. RandREPTILE had higher validation accuracy however it performed slightly worse on the F1 score during testing. These models may need to run for many more update steps to determine if the results improve since the F1 score appears to still be increasing as the model is being run.

An observation was that starting with the pretrained distilBERT using RandEPTILE, the model had a hard time improving the validation accuracy after many update steps. It is possible the model has a hard time learning starting from the initial pretrained model with high noise. A future improvement could involve only adding noise to later layers in the model. Another approach could involve introducing noise gradually later in training once the model has reached local optima.

References

[1] Dani Yogatama, Cyprien de Masson d'Autume, Jerome Connor, Tomas Kocisky, Mike Chrzanowski, Lingpeng Kong, Angeliki Lazaridou, Wang Ling, Lei Yu, Chris Dyer, et al. Learning and evaluating general linguistic intelligence. *arXiv preprint arXiv:1901.11373*, 2019.

[2] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. *CoRR*, abs/1703.03400, 2017.

[3] Rishikesh Jha Trapit Bansal and Andrew McCallum. Learning to few-shot learn across diverse natural language classification tasks. 2019.

[4] Zi-Yi Dou, Keyi Yu, and Antonios Anastasopoulos. Investigating meta-learning algorithms for low-resource natural language understanding tasks. *CoRR*, abs/1908.10423, 2019.

[5] Alex Nichol, Joshua Achiam, and John Schulman. On first-order meta-learning algorithms. *CoRR*, abs/1803.02999, 2018.