FAME-BERT: Stable Meta-learning for Robust Question-Answering

Stanford CS224N Default Project - RobustQA track

Bharat Khandelwal
Department of Computer Science
Stanford University
`bharatkh@stanford.edu`

Shubham Anand Jain
Department of Computer Science
Stanford University
`shubhamj@stanford.edu`

Abstract

With the increasing importance of conversational AI, search engines, and other interactive systems, Question Answering has become a critical task to advancing the state-of-the-art in Natural Language Understanding systems. An important problem is for these systems to generalize well to new domains with a small amount of training examples - understanding how to build such systems will represent a leap of knowledge in our quest for Artificial General Intelligence, and will allow us to create general-purpose software that adapts on an as-needed basis.

To obtain good out-of-domain generalization performance with fewshot learning, we propose our model FAME-BERT (**F**inetune-**A**ugment-**M**etalearn-**E**nsemble Distil**BERT**). We recognize and underline the benefits of carefully crafted learning rates, data augmentation, and ensembling over our base approach of meta-learning a DistilBERT model. Highlights of our model's performance include **Rank 1** by EM on the validation leaderboard and **Top 5** by EM on the test leaderboard. On both validation and test sets, our model outperforms the DistilBERT baseline by a significant margin. Finally, we discuss future directions of research that are likely to further boost model performance.

1 Key Information to include

- Mentor: Fenglu Hong
- External Collaborators (if you have any): N/A
- Sharing project: N/A

2 Introduction

Question-Answering is one of the critical problems in Natural Language Understanding which acts as a milestone for us to measure how well NLP systems today can comprehend context and provide suitable responses. Today, Question-Answering systems are found everywhere: Search engines, Conversational AI, Medical Software, and even in Major League Baseball [1] and health science (EAGLi). However, most of these systems are domain-specific, i.e. they operate in a given context. Given the large amount of data available now, with corpuses such as Wikipedia at our disposal, a natural extension of domain-specific question-answering is to ask whether these models can answer questions on a new domain with a small amount of training data provided at test time. An example prompt of such a question and answer is given in Figure 1.

Meta-learning [2] is a technique that teaches models to "learn how to learn"; more precisely, it focuses on training models on tasks in a way that makes it easier to generalize to completely new tasks, sometimes even with drastically different loss functions, where the pre-training knowledge might be

Question: Why was Tesla returned to Gospic?
Context paragraph: On 24 March 1879, Tesla was returned to Gospic under police guard for not having a residence permit. On 17 April 1879, Milutin Tesla died at the age of 60 after contracting an unspecified illness (although some sources say that he died of a stroke). During that year, Tesla taught a large class of students in his old school, Higher Real Gymnasium, in Gospic.
Answer: not having a residence permit

Figure 1: Example Prompt & Response

helpful. Keeping this visualization in mind, we believe meta-learning is a good approach for Robust Question-Answering, with the training tasks being the in-domain datasets and test tasks being the out-of-domain datasets. Thus, the purpose of this work is to investigate meta-learning solutions for this problem, and brainstorm ideas that can make these solutions more effective.

To obtain good out-of-domain generalization performance with fewshot learning, we propose our model **FAME-BERT** (**F**inetune **A**ugment **M**etalearn **E**nsemble Distil**BERT**). We recognize and highlight the benefits of carefully crafted learning rates, data augmentation, and ensembling over our base approach of meta-learning a DistilBERT model. Our model gives a performance of 53.065 F1 score and 40.314 EM score on the validation leaderboard (Rank 1 by EM) and 60.042 F1 score and 42.959 EM score on the test leaderboard (Top 5 by EM).

3 Related Work

The term “Meta-learning" refers to a variety of methods that guide models *to learn how to learn*; We wish for pre-trained models to learn a large number of new, unseen tasks with minimal training examples and updates. The idea of meta-learning is to have a model that represents a high-level understanding of a domain, which can easily be fine-tuned with minimal updates on any specific task with a small amount of new data.

The paper [2] introduces model-agnostic meta-learning, which is the first work that performs meta-learning without relying on learning rules, updates functions or model architecture.

Suppose we have a pre-trained model that is parameterized by parameter θ. Meta-learning aims to fine-tune θ in a way that allows a small amount of gradient steps to lead to a significant decrease in loss in a large number of tasks; an illustration of how meta-learning uses the same θ to quickly adapt to many tasks can be seen in Figure 2. We talk about more intricate details of meta-learning algorithms in Appendix A.

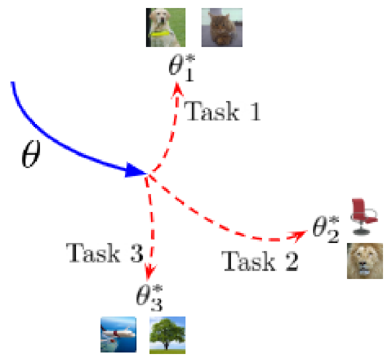


Figure 2: Illustration of Meta-learning

The paper [3] applies the above methods to Natural Language Understanding tasks, in a bid to show that meta-learning performs better than multi-task learning in low-resource language understanding task scenarios. They argue that meta-learning learns a representation that is more unbiased, compared to multi-task learning representations which are biased towards high-resource languages. They evaluate the algorithms on the GLUE [4] dataset, which includes a wide range of tasks such as sentiment analysis, textual entailment, question answering. The authors also try out different ways of choosing the distribution $p(\mathcal{T})$, and claim that the Reptile algorithm with a PPS (Probability proportional to size of task, as compared to, for example, uniform task sampling) distribution tends to work the best.

EDA [5]: Easy Data Augmentation techniques are a set of techniques introduced for better generalisation on tasks with limited resources. It consists of four operations: synonym replacement, random insertion, random swap, and random deletion. An example can be seen in Table 1. We use these synonym replacements in both the contexts and questions for better generalisation of our QA model, so that it doesn't simply learnt to search for the most word matches in question and context.

Operation	**Sentence**
None	A sad human comedy played out on the back roads of life.
Synonym Replacement	A **lamentable** human comedy played out on the **backward** roads of life.
Random Insertion	A sad human comedy played out on **funniness** the back roads of life.
Random Swap	A sad human comedy played out on **roads** back **the** of life.
Random Deletion	A sad human out on the roads of life.

Table 1: EDA Techniques

4 Approach

4.1 Baseline

For the baseline, we use a DistilBERT model [6] given to us as part of the project handout, which is a smaller version of BERT, a pre-trained model. We train it for 3 epochs on the in-domain train data and do not finetune on the downstream out-of-domain tasks.

4.2 Our model

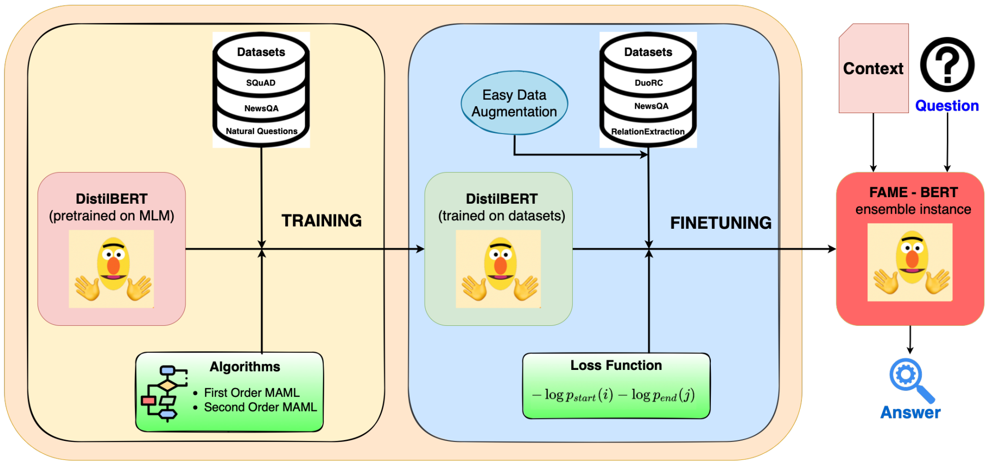


Figure 3: Training Pipeline for FAME-BERT

Our FAME-BERT architecture can be found in Figure 3. Our Training pipeline consists of three major parts:

1. Initial training: In this stage we perform initial training of a pretrained DistilBERT model on the in-domain train datasets of the Question-Answering task. We may either perform

standard training or Meta-update based training, or we may do one on top of the other (for example, a single epoch of MAML over a few epochs of baseline training).

2. Fine-tuning: We perform training on the OOD train datasets, possibly augment them using synonym replacement, and tune Learning rates and number of epochs for best performance on a separately carved out validation dataset.

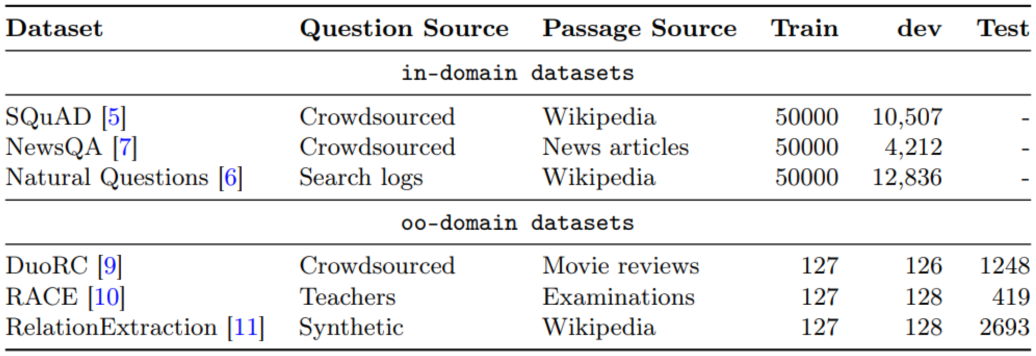
3. Ensembling: We run fine-tuning across multiple seeds and combine the predictions using a majority vote.

Apart from the starter code, we designed the remaining training pipeline ourselves, and implemented Meta-learning algorithms for scratch for our use case. We also designed interfaces for individual learning rates and ensembling of predictions. We adapted code from the original EDA paper [1] to fit to our use case.

5 Experiments

5.1 Data

We use 6 datasets overall, 3 for the training phase, and 3 for the evaluation phase: Three large in-domain reading comprehension datasets (Natural Questions [7], NewsQA [8] and SQuAD [9]) each with 50000 examples, are used as training datasets and three small out-of-domain datasets (RelationExtraction [10], DuoRC [11], RACE [12]) each with only 127 training examples, are used as the fine-tuning datasets. Final evaluation is done on the OOD validation & test datasets. Further details on these datasets are given in Figure 4.



Dataset	**Question Source**	**Passage Source**	**Train**	**dev**	**Test**
		`in-domain datasets`			
SQuAD [5]	Crowdsourced	Wikipedia	50000	10,507	-
NewsQA [7]	Crowdsourced	News articles	50000	4,212	-
Natural Questions [6]	Search logs	Wikipedia	50000	12,836	-
		`oo-domain datasets`			
DuoRC [9]	Crowdsourced	Movie reviews	127	126	1248
RACE [10]	Teachers	Examinations	127	128	419
RelationExtraction [11]	Synthetic	Wikipedia	127	128	2693

Figure 4: Dataset details

5.2 Evaluation method

EM (Exact Match) and F1 scores are used as quantitative evaluation metrics, defined as follows:

- EM score: This is a binary measure of whether the answer is correct. The intuition behind EM is as follows: “Is the given answer exactly the correct one based on the provided answer?"

- F1 score: The F1 score is defined as $\frac{2 \times precision \times recall}{precision + recall}$. The intuition behind F1 is as follows: “How close is the given answer to the provided answer?"

Three human-provided answers are present for each question, and the maximum F1/EM score is taken across these. Then scores are averaged (weighted averaging) over all three evaluation datasets to get the final reported scores.

Model Name	Model Description	Train epochs	Train time (hours)
$\mathcal{M}_1$	DistilBERT baseline, no finetuning	3	4
$\mathcal{M}_2$	DistilBERT baseline with EDA fine-tuning	10	13
$\mathcal{M}_3$	First-order MAML	10	17
$\mathcal{M}_4$	Second-order MAML on $\mathcal{M}_2$	1	5

Table 2: Model descriptions

5.3 Experimental details

The models we choose to compare are as described in Table 2, with the training times provided there as well. The following configurations are common across models in the training phase:

- Number of model parameters: (Base DistilBERT parameters - 66 million)
- Learning rate: AdamW optimizer with learning rate = 3×10^{-5}
- For Meta-learning algorithms, the inner-loop optimizer & learning rate (i.e, the θ_i value) is set to be an SGD optimizer with a learning rate of 3×10^{-3}; this was chosen because a ratio of $100 : 1$ between inner and outer loop values is common practice in some implementations we looked at[2].
- Batch size is 16 for all training steps, and 4 for all fine-tuning steps. The number of epochs trained for depends on the model, as given in Table 2.
- We introduce 2 new training examples for each OOD training example when using data augmentation. The probability of each word being replaced by its synonym is set to be 20%.
- For submission on the test leaderboard, we ensembled over 10 seeds.

5.4 Preliminary Results

Based on Preliminary findings, we had the following observations:

- EDA does not seem to help models during initial training. This is likely because we did not find the correct hyperparameters for EDA to work well; since EDA causes training times to be about 3 times longer (and since the model losses continue to look reasonable for a small subset of data), we were unable to perform a complete hyperparameter search due to heavy compute. We realized that this was too much to ask for, and instead focused our attention on tuning hyperparameters for EDA usage in fine-tuning, which paid off.
- We noted that a single model had vastly different performances v/s number of fine-tuning epochs on the three different datasets. Moreover, the fine-tuning was not very stable, as observed in Figure 6. We realized this was due using a common learning rate despite immense diversity in the characteristics of the underlying data; indeed, after tuning the learning rate and training epochs for datasets individually, we obtained much better performance 5. This is in line with the ideas from [13], which talks about the instability of training meta-learning algorithms.
- From the paper [13], we tried to implement the idea of performing a single stage of second-order MAML on top of a baseline (re-training), which the paper claims to be better and leads to more stable training. This leads to our proposed model $\mathcal{M}_4$. Unfortunately, our $\mathcal{M}_4$ does not end up performing too well; again, this is likely due to lack of a hyperparameter sweep, as we were limited by the compute. A second order gradient is quite heavy in computation; With just a single gradient step in the inner-loop, this takes 4.5 hours per epoch to train[3].

Based on the observations in this subsection, we made many changes to our models. Our detailed analysis in section 6 shows proof via ablation that each of these changes helped.

[1]Code is adapted from here
[2]For example, this one
[3]We implement Second Order MAML based on the library [14] from Facebook Research

5.5 Final Results

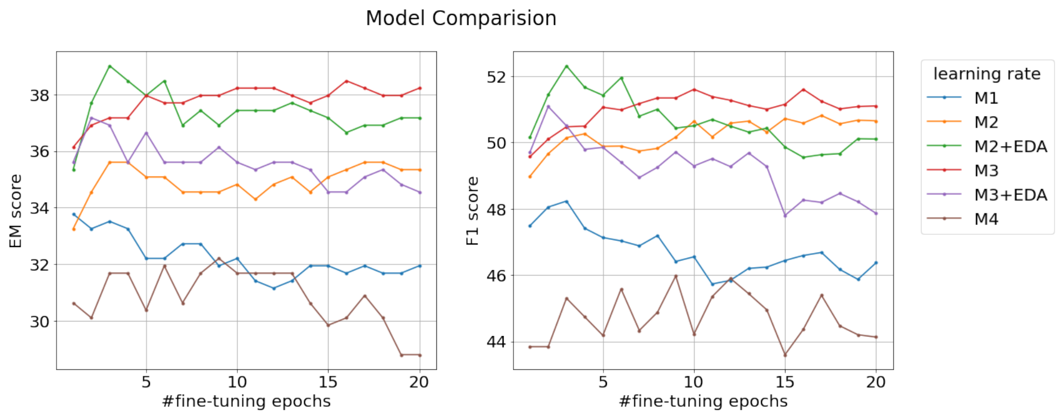


Figure 5: EM and F1 on dev set v/s finetuning steps across models

Model Name	EDA	dev F1	dev EM	Test F1	Test EM
$\mathcal{M}_1$	No	46.512	31.675	59.187*	40.28*
$\mathcal{M}_2$	No	51.995	37.173	-	-
$\mathcal{M}_2$	Yes	**53.020**	**39.791**	**59.679**	42.156
$\mathcal{M}_3$	No	52.128	39.529	59.347	**42.431**
$\mathcal{M}_3$	Yes	51.276	37.958	-	-
$\mathcal{M}_4$	No	46.736	33.770	-	-
Ensemble($\mathcal{M}_2, \mathcal{M}_3$)	Yes	**53.065**	**40.314**	**60.042**	**42.959**
Leaderboard rank (as of writing)	-	**6**	**1**	**11**	**5**

Table 3: Model scores

Our final results can be found in Table 3, and plots for performance of various models on the dev set can be found in Figure 5. Based on our performance on the dev leaderboard, we submitted $\mathcal{M}_2$ + EDA, and $\mathcal{M}_3$ on the test leaderboard. For each of the test submissions, instead of using a prediction of one model, we used an ensemble of models fine-tuned using different seeds. Fine-tuning is a very cheap operation and so this gave us the benefit of getting a set of diverse models and not being forced to use a bad seed. The performance of the ensemble being better than the underlying models on both the dev and test sets was quite surprising, as the ensemble is similar to an averaging metric, and we discuss the possible reasons behind this in next section.

Our final test submission was an ensemble model of $\mathcal{M}_2, \mathcal{M}_3$, which performs best out of all of our submissions, achieving a Test EM score of **42.959** and a Test F1 score of **60.042**, with a dev leaderboard **rank 1** and test leaderboard rank in **top 5** wrt EM at time of writing. The cross-model ensembling criteria used here was motivated by the diverse performance characteristics of $\mathcal{M}_2, \mathcal{M}_3$ on different data-sets.

We did succeed in improving the EM scores quite a bit over the baseline, and one of the major reasons for the success was tuning our learning rates for each datasets when doing meta-learning. While the fact that our dev rank is better than our test rank suggests that we likely overfit the dev data, our dev scores were much lower than test scores, which makes the case that either the dev set was not a good representative sample of the test set or that we didn't overfit a lot; we believe the former to be the case.

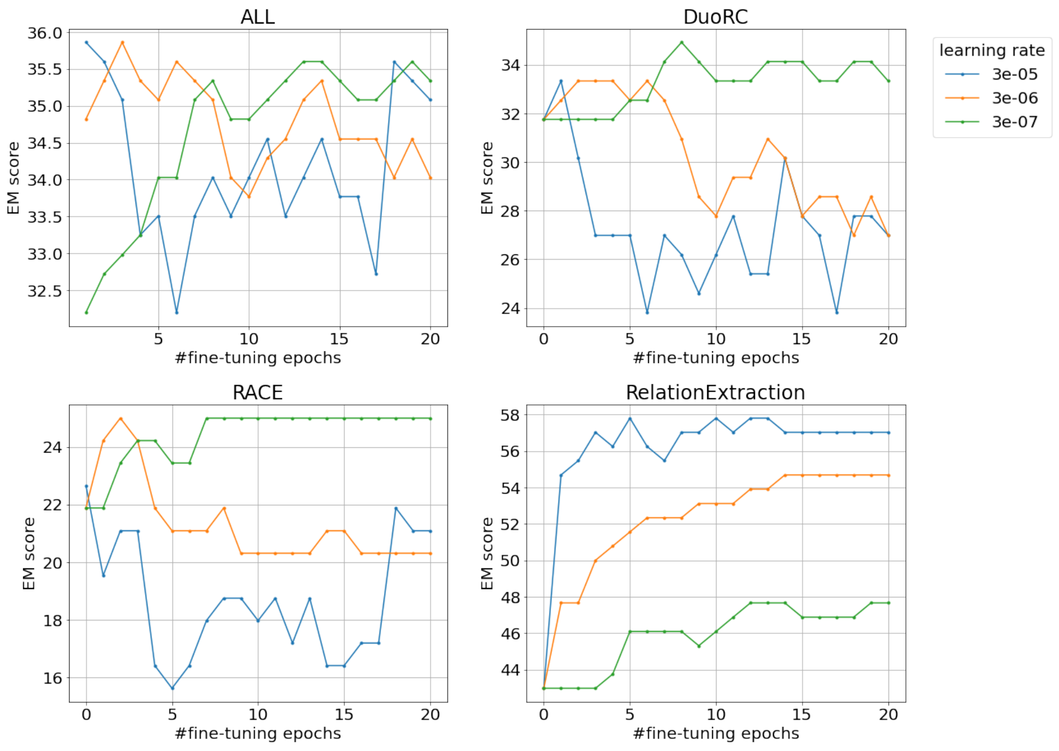


Figure 6: Learning rate instability across datasets

6 Analysis

6.1 Effect of Learning Rates

In Figure 6, we observe the instability of learning rates during fine-tuning across datasets. The correct learning rates, for instance, 3×10^{-7} for DuRoC (Figure 6[b]) and RACE (Figure 6[c]), and 3×10^{-5} for RelationExtraction (Figure 6[d]), are about 100x apart, and if we set them appropriately then the training is much more stable as evidenced by Figure 5. This suggests that we should investigate optimizer variants that can automatically handle this vast diversity in learning rates across datasets. We also realised that there is a lot of scope to improve our model's predictions if we could fit learning rates for each individual dataset during not just fine-tuning but also during initial training, something that we couldn't explore due to long training times and limited compute.

6.2 OOD Dataset Variance

We found that no matter what algorithm we used, there was a lot of variance in terms of our performance on the out of domain datasets. While we performed well on RelationExtraction, the performance of meta-learning on RACE in fact fell in terms of our metric scores, as we increased the number of epochs spent in fine-tuning as shown in Figure 7. This was very counter-intuitive, and we noticed a similar pattern in the baseline model. There is a lot of headroom for improvement here as witnessed by our best performing Ensemble($\mathcal{M}_2, \mathcal{M}_3$) model that leverages best of both worlds by combining better performance of $\mathcal{M}_2$ (baseline) on RACE and better performance of $\mathcal{M}_3$ (meta-learning) on RelationExtraction and DuoRC by combining them through a majority voting mechanism.

6.3 Data Augmentation

After choosing the best learning rate per dataset, we fine-tuned our models, and saw that they had a relative ordering wherein the $\mathcal{M}_3$ was only slightly better than $\mathcal{M}_2$ model when used in conjunction

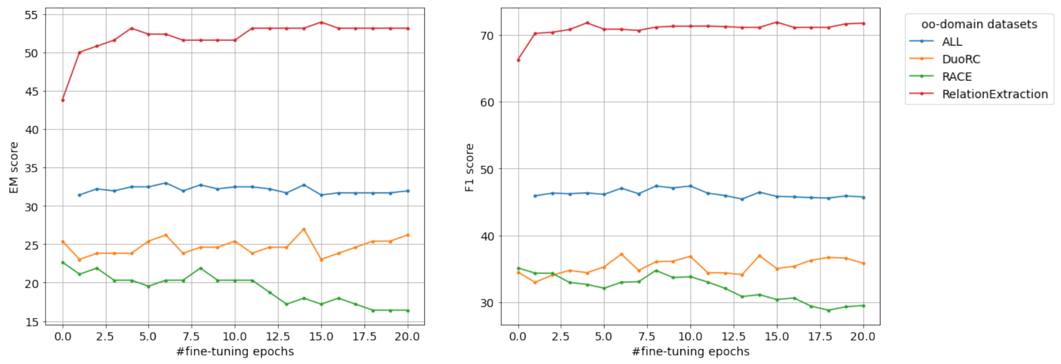


Figure 7: EM and F1 on dev set v/s finetuning steps for $\mathcal{M}_4$ across datasets

with Easy Data Augmentation, as seen in 3. However, EDA didn't seem to help meta learning ($\mathcal{M}_3$) which was a bit surprising. Due to limited time we weren't able to identify the root cause with certainty, but we suspect not choosing the right hyperparameters for data augmentation, and presence of data augmentation during fine-tuning but lack of data augmentation during training steps, to be the reasons behind failure of EDA in improving performance of meta-learned models.

6.4 Ensemble Characteristics

We note that combining predictions across 5 seeds using majority voting for the same model makes a significant difference. To demonstrate this, we show the performance of 5 models without ensembling, and demonstrate the performance boost with ensembling in Table 4.

Model	F1	EM
Average of 5 Seeds	51.11 ± 0.24	38.74 ± 0.29
Ensemble (Majority Vote)	**52.073**	**38.743**

Table 4: Ensemble performance

When we fine-tuned a set of 5 ensemble models starting from different seeds, 88.2% of the time their predictions were identical, and 99% of the times we had a clear majority structure in the vote with more than 50% of the models giving identical outputs. Also, none of the individual models was in a position that it was always correct, i.e., each model's predictions were improved upon by the ensemble and no one underlying model was a clear winner. Although ensembling kept EM scores for the ensemble close to the mean of the underlying models, shown in Table 4, the F1 scores for the ensemble were upto 4 standard deviations higher than the mean of the F1 scores of the underlying models, which is a significant improvement.

7 Conclusion

In this project, we find that meta-learning can indeed outperform vanilla gradient descent based learning methods. We also see the power of ensembling when using the same model across multiple datasets. Using our models, we were able to beat the baseline model by a significant margin and achieve a performance of 53.065 F1 score and 40.314 EM score on the validation leaderboard (Rank 1 by EM) and 60.042 F1 score and 42.959 EM score on the test leaderboard (Top 5 by EM).

There are still a multitude of methods that one could try for future research in this direction, some of which are: PPS sampling by augmenting the in-domain datasets with the OOD datasets and importance sampling, apart from the three main methods that we could not investigate in full detail: learning rate grid search during training, data augmentation hyperparameter search for meta-learning, and stable second-order meta-learning training. We hope these rich plethora of ideas that we leave, inspire future readers to implement them and further improve their models.

References

[1] Bert F. Green, Alice K. Wolf, Carol Chomsky, and Kenneth Laughery. Baseball: An automatic question-answerer. In *Papers Presented at the May 9-11, 1961, Western Joint IRE-AIEE-ACM Computer Conference*, IRE-AIEE-ACM '61 (Western), page 219–224, New York, NY, USA, 1961. Association for Computing Machinery.

[2] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In Doina Precup and Yee Whye Teh, editors, *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 1126–1135. PMLR, 06–11 Aug 2017.

[3] Zi-Yi Dou, Keyi Yu, and Antonios Anastasopoulos. 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, Hong Kong, China, 2019. Association for Computational Linguistics.

[4] Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 353–355, Brussels, Belgium, November 2018. Association for Computational Linguistics.

[5] Jason Wei and Kai Zou. Eda: Easy data augmentation techniques for boosting performance on text classification tasks. *arXiv preprint arXiv:1901.11196*, 2019.

[6] Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter, 2020.

[7] Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, Kristina Toutanova, Llion Jones, Matthew Kelcey, Ming-Wei Chang, Andrew M. Dai, Jakob Uszkoreit, Quoc Le, and Slav Petrov. Natural questions: A benchmark for question answering research. *Transactions of the Association for Computational Linguistics*, 7:452–466, 2019.

[8] Adam Trischler, Tong Wang, Xingdi Yuan, Justin Harris, Alessandro Sordoni, Philip Bachman, and Kaheer Suleman. NewsQA: A machine comprehension dataset. In *Proceedings of the 2nd Workshop on Representation Learning for NLP*, pages 191–200, Vancouver, Canada, August 2017. Association for Computational Linguistics.

[9] Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. SQuAD: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2383–2392, Austin, Texas, November 2016. Association for Computational Linguistics.

[10] Omer Levy, Minjoon Seo, Eunsol Choi, and Luke Zettlemoyer. Zero-shot relation extraction via reading comprehension. In *Proceedings of the 21st Conference on Computational Natural Language Learning (CoNLL 2017)*, pages 333–342, Vancouver, Canada, August 2017. Association for Computational Linguistics.

[11] Amrita Saha, Rahul Aralikatte, Mitesh M. Khapra, and Karthik Sankaranarayanan. DuoRC: Towards complex language understanding with paraphrased reading comprehension. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1683–1693, Melbourne, Australia, July 2018. Association for Computational Linguistics.

[12] Guokun Lai, Qizhe Xie, Hanxiao Liu, Yiming Yang, and Eduard Hovy. RACE: Large-scale ReAding comprehension dataset from examinations. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 785–794, Copenhagen, Denmark, September 2017. Association for Computational Linguistics.

[13] Antreas Antoniou, Harrison Edwards, and Amos J. Storkey. How to train your MAML. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019.

[14] Edward Grefenstette, Brandon Amos, Denis Yarats, Phu Mon Htut, Artem Molchanov, Franziska Meier, Douwe Kiela, Kyunghyun Cho, and Soumith Chintala. Generalized inner loop meta-learning. *arXiv preprint arXiv:1910.01727*, 2019.

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

A Appendix A

The goal of meta-learning is teaching models to learn how to learn. Concretely, assume that we have a (possibly diverse) set of tasks $\mathcal{T}$, where the output for the i^{th} task is given by f_θ, the loss function being given as $\mathcal{L}_i$. We are also given a probability distribution for sampling over these tasks $p(\mathcal{T})$. Model-agnostic meta learning works as follows:

1. Consider a pre-trained model $\mathcal{M}$ with parameters θ, which has a large corpus of knowledge, such as DistilBERT.
2. We update the parameters θ to be a good representation to easily learn new tasks; that is, θ is "sensitive" to new updates. This is called the meta-update step.
3. Optionally, we may perform re-training (for example, training DistilBERT on Question-Answering and re-training with meta-learning) which may help with training stability [13].
4. During test time, we fine-tune θ on the target task that has a small number of training examples (few-shot learning).

The crucial part of meta-learning is the meta-update step. In this step, we wish θ to satisfy

$$\min_\theta \sum_{\mathcal{T}_i \sim p(\mathcal{T})} \mathcal{L}_i\left(f_{\theta_i}\right),$$

where $\theta_i = \theta - \alpha \nabla_\theta \mathcal{L}_i(f_\theta)$ is the result of applying a single gradient step on the i^{th} task (we can also apply multiple gradient steps and meta-learn α; this does not change the core idea). This minimization forces θ to be a **good central representation** that learns well across tasks $\mathcal{T}$, and thus we may expect it to learn quickly on a new, unseen task. To perform this minimization, [2] suggests the MAML algorithm which performs the stochastic gradient update:

$$\theta \leftarrow \theta - \beta \nabla_\theta \sum_{\mathcal{T}_i \sim p(\mathcal{T})} \mathcal{L}_i\left(f_{\theta_i}\right)$$

Note that this algorithm involves the *gradient through a gradient*; θ_i itself is a gradient descent update on θ. This is an expensive operation (indeed, in section 5 we note that it makes training around $3\times$ slower); to mitigate this, [2] also proposes a first-order approximation to this, called First-Order MAML. To address this issue of computation time in another manner, the paper [15] presents Reptile, which provides an alternate linear update method for θ.