Examining the Effectiveness of a Mixture of Experts Model with Static Fine-tuned Experts on QA Robustness

Stanford CS224N Default Project (Robust QA)

Matthew Kolodner
Department of Computer Science
Stanford University
`mkolod@stanford.edu`

Jack Xiao
Department of Computer Science
Stanford University
`jackxiao@stanford.edu`

Abstract

While much progress has been made in recent years on modeling and solving natural language understanding problems, these models still struggle to understand certain aspects of human language. One of the most difficult areas for current models is generalization. While humans can easily generalize beyond a training data set, computers often have difficulty developing non-superficial correlations beyond the provided data. In this project, we tackled this concept of computer generalization through the development of a robust question answering (QA) system that is able to generalize answers to questions from out-of-domain (OOD) input. Here, we applied a modified Mixture of Experts (MoE) model, where gating and expert training are handled seperately, over the 6 datasets in order to create robustness through specialization of the various expert models. We also applied few-sample fine-tuning to large and small components of the model to try to better account and generalize for cases where there is little data. Ultimately, from the results of the model, we observed that this modified MoE architecture has several limitations through its expert and training method and was unable to improve significantly on the baseline of the model. In addition, we also observed that the few-sample fine-tuning techniques greatly improved the performance of the small, out-of-domain expert but barely improved, and sometimes harmed, models with a larger dataset. As a whole, this paper illustrates the potential limitations of applying a simple MoE model and few-sample fine-tuning to the complex task of generalization and may suggest the implementation of more advanced structures and techniques are necessary for strong performance.

1 Key Information to include

- Staff Mentor: Yuyan Wang

2 Introduction

In recent years, we have seen a gradual improvement in ability for computers to model and solve natural language understanding problems. This can be seen through the implementation of various state-of-the-art architectures such as BERT [1]. Yet, although certainly much progress had been made in this field, there is still much work to be done for computers to be able to fully understand human language [2] [3] [4] [5]. One current area that many models still struggle with is applying learned characteristics to out-of-domain data through generalization. While humans are typically able to generalize learned information beyond a training data set, computers often have a much more difficult time developing important correlations beyond data that it has been taught. There is a strong need for computer models to identify non-superficial relations as they learn in order to achieve a more robust performance as a whole. In this paper, we explore a method for improving computer generalization through the development of a robust QA system through various model architectures and techniques.

3 Related Work

3.1 Mixtures of Local Experts

Mixture of Experts (MoE) is a model that utilizes several specialized experts in order decompose and improve performance on specific sub tasks within a model [6]. This model also utilizes a gating function in order to assign experts to their corresponding subtasks. The resulting modified measure of error in the MoE model allows the individual experts to remain mostly localized in the error calculation, better optimizing for their own subtasks.

Rather than weighting the experts and comparing them collectively to the desired output, each of the experts seeks to optimize of the desired output independently. Although there may still be some impact between experts, this effect is relatively small compared to prior implementations of similar models, as the sign of the error and update for each expert will still be accurate and localized. Furthermore, this will force expert that performs worse than the weighted average of all the experts to receive less priority from the gating network, while an expert that performs better than the weighted average of all the experts will receive more priority. By the end of training, this will ensure that each expert is specialized in their respective field.

This model has already been successfully applied this to a 4-class vowel discrimination problem. The model and results discussed in [6] suggest that applying MoE as a means of decomposing a problem may be applicable in a larger context. We hope to take this MoE framework and apply it to our problem of developing a robust QA model. Much like how experts were able to be connected to several subtasks in the paper, we also hoped to divide the QA problem into subtasks based on the datasets used for training the model. This way, out-of-domain data would be more accurately identified and modeled through an out-of-domain specialized expert.

3.2 Few-sample BERT Fine-tuning

Several approaches are described in [7] for improving the performance of pre-trained language models where there is a very small language model with a small amount of training data. One of the techniques discussed in paper suggests including the debiasing step in ADAM in order to achieve better performance overall. Another technique suggests re-initializing the top layers of a pretrained model, since these layers typically encode specialized information specific to the pretraining dataset. This way, the pretraining can perform better on the small dataset. Another method discussed suggests fine-tuning the model for longer periods of time in order to improve training stability and model performance. Several other existing methods have also shown promise, including utilizing pre-trained weight decay, mixout (replacing a parameter in the model with the corresponding pre-trained parameter with probability p at each iteration), and layer-wise learning rate decay. After applying these techniques to several problems, the paper deduces that utilizing these techniques can greatly improve performance when using a few-sample data set. In our paper, we hope to apply some of these techniques to a baseline DistilBERT [8] model and our individual MoE experts. These few-sample fine-tuning methods may allow us to significantly improve model performance even with limited data.

4 Approach

4.1 Mixture of Static Experts

Our primary method of approaching this problem was to employ a modified Mixture of Experts model to the datasets provided in order to improve the model's performance for robust question answering [6]. In the MoE model, our goal is to train n experts on data from the model, with each expert being in charge of some specific sub-task of the problem. In the case of RobustQA, we looked to decompose the model's datasets into the n sub-tasks for each expert to train on. In this case, with 3 larger in-domain datasets and 1 small out-of-domain (OOD) dataset, we would have 4 experts, 1 or each of the in-domain datasets and 1 for the entire OOD dataset (since the individual OOD datasets are very small). Unlike in traditional MoE models where experts are trained alongside the gating network, we trained and fine-tuned our experts independently prior to the training of the gating network. In other words, we utilize a Mixture of Experts model with static experts. For each of our 4 experts, we fine-tuned a pretrained DistilBERT model on their respective dataset(s) so they can become specialized in their task. Through this, we control which specific examples update which specific experts and combine the results of the individual experts with a gating function, potentially achieving a greater robustness in our model.

We fine-tuned each of the experts in this modified format, separately from the gating portion of the MoE model, as to avoid memory errors during training while also ensuring a specific assignment of each expert and breakdown of tasks. After completing the fine-tuned model, we looked to train a gating function through a simple multilayer perceptron (MLP) in order to enable the MoE model to select which expert to use for which training example. For our MLP, we initially designed it as a 3 layer softmax gating function consisting of an input layer, an hidden layer, and an output layer. Each layer was calculated using $f(W_iX_i^T + b_i)$, where W_i, b_i, and X_i are the weights and bias and inputs, respectively, for layer i, and f is a ReLU activation function for non-linearity, followed by a softmax applied to the final layer to obtain the output as a probability distribution. We also tested variations of this model, including the removal of the bias term, (with layer calculation formula becoming $f(W_iX_i^T)$) and additional hidden layers. For each training example x, we calculated the output as

$$y = \sum_{i=1}^{n} \mu_i g_i$$

where μ_i is the output of x evaluated on expert i's fine-tuned DistilBERT model and g_i is the output of the gating function applied to x. We evaluated loss L for each example x as

$$L = -\log \sum_i g_i e^{\frac{1}{2}(C_i)}$$

where C_i is the cross entropy loss calculated from x evaluated on expert i's fine-tuned DistilBERT model. One important note is each expert only updates its model parameters in the fine-tuning step on its specific datasets and not in the MoE model computations. This is so the experts can remain specialized and the out-of-domain experts won't overfit to the in-domain datasets, since there is a lot more in-domain data.

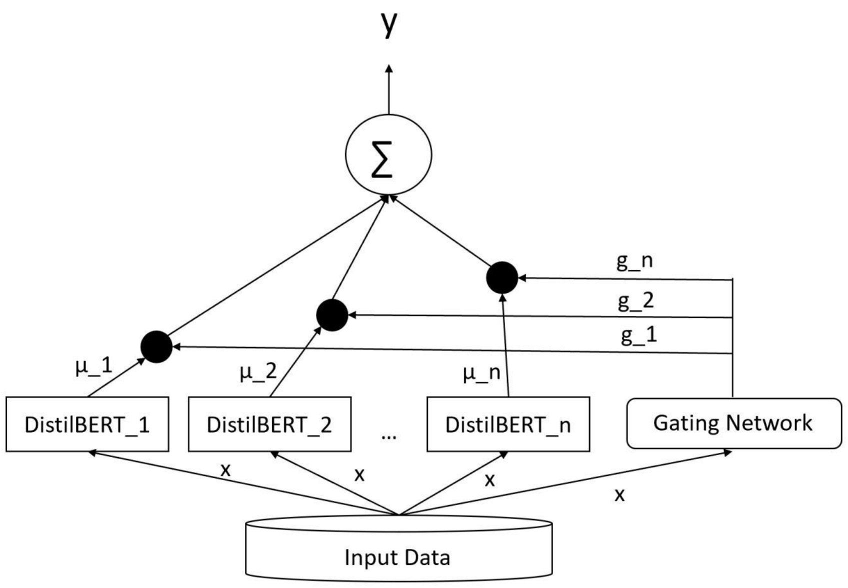

Figure 1: Illustration of DistilBERT MoE Architecture [9]

4.2 Baseline and Few-sample Fine-tuning

For our baseline, we use a DistilBERT model trained on the entire in-domain training set. DistilBERT, which is a smaller, faster version of the larger BERT model while still retaining much of the language understanding capability [8], provides a solid foundation from which we can evaluate potential model improvements.

After applying the MoE model to our dataset, we implemented several few-sample fine-tuning optimizations on both the original DistilBert and on the lone OOD expert. The few-sample techniques we experimented with include longer fine-tuning, pre-trained layer re-initialization, and employing weight decay. Through these methods, we can examine the impact of such few-sample fine-tuning techniques on our baseline DistilBERT model as a whole as well as its impact (as intended in [7]) on performance after training on the very limited OOD dataset.

5 Experiments

5.1 Data

The three in-domain reading comprehension datasets that we will use to train our QA system are SQuAD, NewsQA, and Natural Questions. Our QA system will be tested on three out-of-domain datasets: DuoRC, RACE, and RelationExtraction. From each of the three in-domain datasets, wedraw 50000 training examples. We also draw a small set of 127 training examples from each theout-of-domain datasets. The resulting total training set is of size 150381. In addition to the training set, we also have a dev set consisting of 10507 examples fromSQuAD, 4212 examples from NewsQA, 12836 examples from Natural Questions, 126 examplesfrom DuoRC, and 128 examples from each of RACE and RelationExtraction. The held-out test setconsists of 4360 examples from the three out-of-domain datasets. Each example from the datasets is a triple consisting of a context, a question, and an answer.

5.2 Evaluation method

Our QA system will be evaluated using a the Exact Match (EM) score and the F1 score. These metrics provide a clear comparison between our baseline DistilBERT model, the MoE model, and models refined with few-sample fine-tuning in their ability to select accurate answers from text. Because the focus of our experiment is to improve robustness and performance on OOD datasets, we will primarily use the scores of the models on the OOD validation set for consistency in evaluation.

5.3 Experimental details

Each of our models was trained using an Adam optimizer with $\beta_1 = 0.9$, $\beta_2 = 0.999$, and a learning rate of 3×10^{-5}. Specifically for our MoE models, the input layer for our gating network has a size of 384, based on input sequence length of the data, and our output layer has a size of 4, based on the number of experts. The hidden layer is of size 260, was deduced from taking $\frac{2}{3} \times input\_size + output\_size$ [10]. The majority of training was performed using a batch size of 16, but a batch size of 14 was used when training our MoE experts in order to avoid memory errors.

For our MoE models, we tested different versions of our gating network, which involved varying the number of hidden layers and removing bias terms. Then, to explore few-sample fine-tuning methods, we varied the training epochs (between 3 and 4 when training on the large in-domain dataset and between 3 and 20 when training on the small OOD dataset), and examined the individual/combined effects of layer re-initialization and weight decay. Additionally, by focusing few-sample fine-tuning techniques specifically on the OOD expert, we can get a better idea of the effectiveness of such methods in true few-sample contexts compared to their use in a broader training process on larger datasets.

5.4 Results

Both MoE models performed worse than expected, with scores lower than the basline DistilBERT model (see Table 1). Interestingly, the performance of the two MoE models exactly matched the performance of two individual experts. One of our MoE models matched the results of our SQuAD expert, and the other MoE model matched the results of our OOD expert. This (as confirmed in our analysis) was a result of the MoE system latching onto a single expert and using only that expert for every example input.

MoE Scores (on OOD dev set)		
Expert	F1 Score	EM Score
DistilBERT (baseline)	**48.43**	**33.25**
MoE	43.01	26.18
MoE (with added bias term in gating function)	27.08	18.32
SQuAD Expert	43.01	26.18
Natural Questions Expert	36.79	20.42
NewsQA Expert	39.42	25.39
OOD Expert (20 Epochs)	27.08	18.32

Table 1: Baseline score on OOD dev compared to MoE score and individual expert scores

When implementing few-sample fine-tuning techniques when training on the large in-domain datasets, we were able to observe minor performance improvements (see Table 2). **Through these fine-tuning adjustments, we obtained our best performing model (DistilBERT with 4 epochs of fine-tuning on in-domain data and additional tuning on OOD data), which achieved an F1 score of 58.101 and and EM score of 39.151 on the Robust QA test leaderboard.** Overall, however, the fine-tuning techniques described in [7] provided minimal improvement over the baseline DistilBERT model.

Model Scores (on OOD dev set)		
Model	F1 Score	EM Score
DistilBERT (baseline)	48.43	33.25
DistilBERT (with additional tuning on OOD data)	48.95	33.51
DistilBERT (longer fine-tuning, 4 epochs)	**49.97**	33.77
DistilBERT (longer fine-tuning and OOD data)	49.89	**34.29**
DistilBERT (1 re-initialized layer)	46.87	31.41
DistilBERT (2 re-initialized layers)	46.46	31.15
DistilBERT (with weight decay)	47.00	31.68

Table 2: Baseline compared with fine-tuning techniques

We observe the most interesting and significant results (as we expected) when applying the few-sample fine-tuning techniques on the OOD Expert, tuning only on the small OOD training set. Simply by training longer, we achieve a nearly 6% performance increase. Pretrained layer re-initialization and weight decay provided further performance improvements. Both a 1 layer re-initialization and a 1 layer re-initialization combined with weight decay yield a performance increase of nearly 9% over the baseline Expert trained for 3 epochs, and a 3% increase over the model trained for 20 epochs. Most notably, while training for longer resulted in better performance over all 3 individual OOD datasets, layer re-initialization caused a sharp increase in performance specifically for the DuoRC dataset.

Few-sample Fine-tuning F1 Results for OOD Expert (on OOD dev set)				
Model	DuoRC	RACE	RE	Overall
DistilBERT (3 epochs)	4.35	5.47	54.44	21.51
DistilBERT (20 epochs)	7.52	10.44	62.98	27.08
DistilBERT (1 layer re-init, 20 epochs)	16.18	11.16	62.62	30.06
DistilBERT (2 layers re-init, 20 epochs)	13.74	**11.29**	61.24	28.73
DistilBERT (with weight decay, 20 epochs)	13.08	10.88	61.83	28.68
DistilBERT (with 1 layer reinit + weight decay)	**16.40**	11.16	**63.14**	**30.31**

Table 3: Few-sample fine-tuning on limited OOD data

6 Analysis

6.1 Mixture of Experts Analysis

Even after tuning our hyper-parameters for the MoE model, including adjusting hidden size and number of hidden layers, the baseline model continued to outperform our MoE model. This is made apparent through the higher F1 and EM score of the baseline model compared to both the MoE and the experts. One key observation, as noted earlier, is that both MoE models had an identical F1 and E1 score to two of the experts. More specifically, the MoE model with a weight and bias activation had the same performance as the out-of-domain expert, and the MoE model with just a weight activation had the same performance as the SQuAD expert. Looking into the probability tensor in the model which encodes the selection of expert(s) for each example in a batch, we can see why this is the case. For the MoE model with just a weight activation, this tensor becomes:

$$\begin{bmatrix} 1.00 & 0.00 & 0.00 & 0.00 \\ 1.00 & 0.00 & 0.00 & 0.00 \\ \vdots & \vdots & \vdots & \vdots \\ 1.00 & 0.00 & 0.00 & 0.00 \end{bmatrix}$$

Here, the first column represents the first expert, which is the SQuAD expert, and each row represents one example in a batch. A similar observation was made for the MoE model with a weight and bias activation for the out-of-domain expert. Ultimately, rather than selecting a combination of experts for a specific task or varying the probability of choosing an expert to yield better overall performance results, the MLP instead chooses to only identify and use one expert for the entire problem. In the case of the MoE with only weights, it likely chose the SQuAD expert because it was the best-performing expert. One possible reason why the MoE selecting the out-of-domain expert with the bias parameter could be because the bias parameters were encoding heavy favoritism towards out-of-domain expert because it may perform very well on training, but not extrapolate well to the validation set due to over fitting from the limited training size of the out-of-domain expert.

Either way, this indicates that our MoE model can only perform as well as its best-performing expert, and since each of our experts performed under the baseline, it makes sense that the MoE model as a whole also would do the same. As a result, our MoE model is limited in two ways: firstly, if the previously trained experts are poor-performing (especially for the Out-of-Domain expert), our model will perform poorly. However, in addition, the gating function of our modified MLP also poses a weakness since it can only select one expert. One likely reason why our MoE model observes this secondary weakness is because of the modified nature of the expert model, where the experts are all independently fine-tuned prior to the training of the gating function, and remain static while the gating network is trained. While this does accomplish the task of making sure that only certain subsets of data update certain experts, this also means that we are evaluating the model based on the performance of the each of the experts alone. Ideally, the way the models' performance should be evaluated is based on both the decision of the gating function in addition to the performance of the model, with updates to both the gating network and localized experts during training. This distinction explains why our model only trains on one expert models' performance and why the performance may have fallen short in that regard. However, at the same time, [6] explains that one weakness of their model was that only 3 of the 8 experts were active in the final model. This may indicate that this may be a shortcoming of a primitive mixture of experts approach as a whole, and more work needs to be done in improving the MoE design.

6.2 Few-sample Fine-tuning Analysis

Seeing as our MoE model was largely dependent on its best-performing expert, we decided to conduct a small ablation study to analyze the effect of few-sample fine tuning on both a DistilBERT trained on the entire dataset and the out-of-domain expert. For the DistilBERT trained on the whole model, we observed that only two methods improved the performance of the model: increased time fine-tuning and additional tuning on OOD data, together increasing the F1 performance of the baseline model by 1.54%. On the other hand, each of the other few-sample fine-tuning techniques slighty hurt the model's performance. The likely explanation for this is due to the the relatively large size of the data available. While these techniques were suggested for few-sample performance improvements, in the case of a data set with lots of data, using few-sampling techniques alone won't be able to improve performance on the out-of-domain components of the data which are few in number. Looking at some selected examples from the model, we can deduce specific areas where the few-sample fine-tuning didn't have much effect.

We noticed a potential correlation between context length and accuracy of the prediction. A widespread trend we observed among the output data was that questions with longer corresponding contexts often produced an incorrect result, while questions with shorter corresponding contexts often yield correct results.

Context: As elderly people go about their day in Manhattan, Harry walks along a sidewalk with his tabby cat Tonto on a leash, quoting Shakespeare's "King Lear" as he goes [. . .] Harry meets his old friend Jacob on a bench, and tells him that his apartment building is being torn down to build a parking lot [. . .] Harry soon learns that he can't go through security with Tonto, so he takes a cab to get a bus. Along the route, Harry asks the bus driver to stop so that Tonto can relieve himself, whereupon the cat runs away across a cemetery [. . .]
Context Length: 3757
Question: Who is Harry's travelling companion?
Correct Answer: Tonto
Model Answer: Jacob on a bench, and tells him that his apartment building is being torn down to build a parking lot. Jacob

---

In the above example, we can see that the phrase "travelling companion" is never explicity mentioned in the context, and answering the question correctly would involve deducing the meaning of those question words and understanding the parts of the context that involve traveling and companionship. This can mean that our model had difficulty extracting long-term dependencies as well as identifying a correct prediction in a large context body. Contrast this with a shorter, more simple example:

Context: Stephen Silvagni (born 31 May 1967) is a former Australian rules footballer for the Carlton Football Club.
Context Length: 106
Question: What was the name of Stephen Silvagni's team?
Correct Answer: Carlton Football Club
Model Answer: Carlton Football Club

Upon closer inspection, we might draw the conclusion that length alone is not the root cause behind incorrect predictions. As a whole, we find that longer contexts involve more challenging relationships between words, longer dependencies, and overall more vague and inspecific language (e.g. the use of synonyms, pronouns, lack of shared words between question and context). While few-sample fine-tuning often yields improved early performance from a pretrained model, in this case, we can see that this improved early performance may have little effect in improving long-term dependency extraction on large models. As a result, few-sampled fine-tuning has limited effects on improving a large DistilBERT model.

In addition to the large DistilBERT model, we also experimented with few-sample fine-tuning on our Out-of-Domain expert. We did this because the OOD expert was the worst-performing expert due to its limited datasize and could benefit greatly from few-sample fine-tuning. From our results, we see that implementing several of these fine-tuning techniques on this expert lead to significant improvements in the model's performance, gaining 8.8% from its original F1 score.

One interesting aspect about the data is that this expert performed extremely well on the Relation Extraction dataset for all variations, having at least a 54% F1 score in all cases. Investigating this dataset, one probable reason for this becase this dataset contains many short contexts and simple reading comprehension questions [11]. In contrast, the initial F1 performnce on the DuoRC dataset was very low, at 4.35%. After applying the few-sample improvements through re-initialization of top layers, weight decay, and longer fine-tuning time, the DuoRC dataset was able to see the largest increase in performance, increasing by 12.35%. The DuoRC dataset is known to have more complicated dependencies and and challenging answers considering that the contexts from which and answer is to be predicted and the contexts from which the true answer is derived share little lexical overlap [12].

As a result, it makes sense that the re-initializing of top layers helps, since one key benefit of this technique is the fact that the model can learn more complex dependencies faster. This is built off the intuition that the top layer of the pretrained data encodes specific information for the pretrained data rather than our training data and is better off reinitialized. While this did lead to a very slight decrease in performance on the Relation Extraction dataset, combining this re-initializing feature with weight decay lead to the largest overall improvement. It is likely that re-initializing 2 layers harmed performance compared to 1 layer re-initialization because of how the 2nd layer may have encoded valuable information from pre-training that is less task-specific. That being said, these techniques such as layer re-initialization certainly yield the effects of improved performance, lower training loss, and quicker convergence that [7] describes (see Figures 2 and 3 in Appendix).

Overall, these results illustrate that the few-sample fine-tuning performs well on the out-of-domain expert due to its small data size. Between both the pretrained models that few-sample tuning was performed on, it can be inferred that few-sample tuning works best on small-sample data sets, but could be detrimental to large-sample data sets. In addition, few-sample tuning has difficulty separating small OOD inputs from large in-domain inputs when trained together, as shown by the decrease in F1 performance.

7 Conclusions and Future Work

In this project, we implemented, evaluated, and analyzed a MoE model with few-sample fine-tuning in order to improve QA robustness on out-of-domain questions and contexts. This was our first attempt at implementing a complex deep learning model from scratch, and we both gained a lot of valuable experience in debugging, fine-tuning, and redesigning the implementation. First, we applied a modified MoE model using 4 fine-tuned DistilBERT experts and a MLP gating function. Then, we applied few-sample fine-tuning to both the whole domain DistilBERT model and to the out-of-domain expert in an effort to increase the robustness of these models. With the fine-tuning, we tested several methods including longer training, re-initializing of top-layers, and weight decay.

From the results, we could see that the static MoE model did not perform very well on the out-of-domain data, only being able to perform as well as its best expert while being unable to surpass the baseline model in performance. This is likely due to the limitations imposed by the expert performance and static nature of the experts, although there are also larger considerations of poor performance when applying a simple MoE architecture for complicated problems such as generalization. We also observed that only a few techniques for few-sample fine-tuning had positive effects on the performance of training sets with lots of in-domain data and relatively little out-of-domain data. On the other hand, many of these techniques led to positive results for improving performance of the out-of-domain expert, which had little training data.

In the future, there are many things we hope to do in order to continue improving our implementation. One area of improvement is with the MoE model itself. We had used a static, modified MoE approach in this paper due to memory issues on the GPUs, so one thing we could research could be using the memory more efficiently. This way, we could implement a MoE model that would be able to train its experts at the same time as the MLP gating function. Another thing we could look to do is to apply a different, more advanced MoE architecture to the model, such as a Hierarchical MoE architecture. This experiments with adding multiple gating functions in the selection process, which could lead to more experts being selected, as opposed to a single expert being chosen for every example. We could also improve on the few-sample fine-tuning by investigating the effect other techniques that we did not have time to test and implement, such as mixout and layer-wise learning rate decay.

References

[1] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding, 2019.

[2] Robin Jia and Percy Liang. Adversarial examples for evaluating reading comprehension systems, 2017.

[3] Suchin Gururangan, Swabha Swayamdipta, Omer Levy, Roy Schwartz, Samuel R. Bowman, and Noah A. Smith. Annotation artifacts in natural language inference data, 2018.

[4] R. Thomas McCoy, Ellie Pavlick, and Tal Linzen. Right for the wrong reasons: Diagnosing syntactic heuristics in natural language inference, 2019.

[5] Marco Tulio Ribeiro, Tongshuang Wu, Carlos Guestrin, and Sameer Singh. Beyond accuracy: Behavioral testing of nlp models with checklist, 2020.

[6] Robert A. Jacobs, Michael I. Jordan, Steven J. Nowlan, and Geoffrey E. Hinton. Adaptive mixtures of local experts. *Neural Computation*, 3(1):79–87, 1991.

[7] Tianyi Zhang, Felix Wu, Arzoo Katiyar, Kilian Q. Weinberger, and Yoav Artzi. Revisiting few-sample bert fine-tuning, 2020.

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

[9] Jongwon Yoon, Sung-Ihk Yang, and Sung-Bae Cho. Adaptive mixture-of-experts models for data glove interface with multiple users. *Expert Syst. Appl.*, 39:4898–4907, April 2012.

[10] Jeff Heaton. The number of hidden layers, Jun 2017.

[11] Omer Levy, Minjoon Seo, Eunsol Choi, and Luke Zettlemoyer. Zero-shot relation extraction via reading comprehension, 2017.

[12] Amrita Saha, Rahul Aralikatte, Mitesh M. Khapra, and Karthik Sankaranarayanan. Duorc: Towards complex language understanding with paraphrased reading comprehension, 2018.

8 Appendix

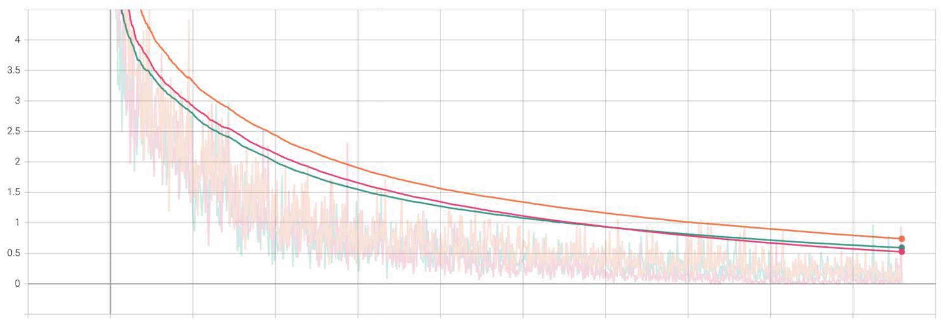

Figure 2: Training loss of OOD Expert. Notice that the green line (fine-tuning with 1 layer re-initialization) is consistently lower than the orange line (baseline).

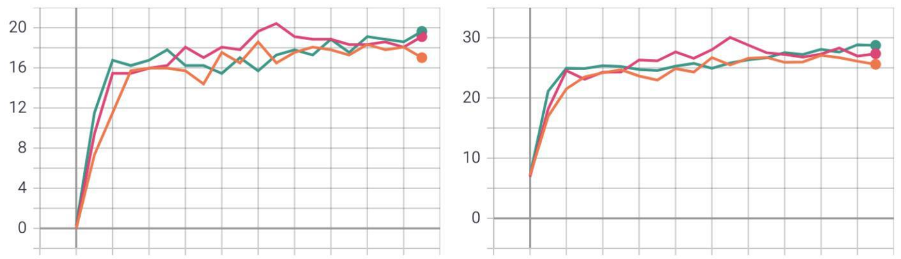

Figure 3: EM (left) and F1 (right) scores of OOD Expert on OOD dev set. Notice that the green line (fine-tuning with 1 layer re-initialization) rises quicker and converges above the orange line (baseline).