A Study on Adversarial Training for Robust Question Answering

Stanford CS224N Default Project
Track: RobustQA

Laura Fee Nern
Department of Computer Science
Stanford University
lfee590@stanford.edu

Kathleen Pietruska
Department of Computer Science
Stanford University
kapie102@stanford.edu

Abstract

Question Answering is a well-known and frequently studied topic in Natural Language Processing. An ideal question answering model does not concentrate on internal particularities of the training data which might not be relevant for the general task of reading comprehension, and instead generalizes well to unseen datasets. A suitable concept that helps increasing domain invariance of a model is adversarial training. In this report, we consider three different approaches for adversarial training and compare them empirically. We conduct an extensive study, where we fine-tune each of the approaches, in order to investigate which methodology is most effective in promoting domain adaption of a model on the question answering task. Our results indicate that adversarial training incorporating a discriminator to challenge the question answering model is most promising.

1 Key Information to include

- Mentor: Sarthak Kanodia
- External Collaborators (if you have any): -
- Sharing project: -

2 Introduction

Question Answering (QA) is an established application in the field of NLP and a lot of progress has already been made, e.g., [1, 2, 3, 4]. A QA model tries to solve a reading comprehension task by identifying the correct text span of a passage that provides the answer to a given question. This helps extracting relevant information from a text to answer a users question and could be, for example, incorporated into a chatbot to answer a customers question in a very user-friendly way. One could further combine it with a knowledge retriever that provides the QA model with a relevant text from a database to answer a specific question.

A main improvement on this task was possible by using huge pretrained transformer based language models and fine-tuning them on QA, as suggested, e.g., for BERT, an bidirectional encoder model, in [3]. Today, almost all state-of-the-art QA systems are based on pretrained transformer models. Modern QA systems are fast and perform accurate on datasets like SQuAD [1] when tested and trained on examples from the same data distribution.
However, these QA systems tend to overfit to seen in-domain datasets and do neither adapt to less seen out-of-domain data, nor generalize well to completely new datasets [5]. It seems that these models do not learn the overall task in a general way but adapt to inherent brittle structures of the data that are expressive for the specific training set only. When deploying such a model

in production or testing it on a new dataset, the impressive performance collapses reducing its benefit in real world systems.

To promote common use of such models, robust QA systems which generalize well across different domains without further explicit fine-tuning are required. There is a lot of research studying techniques to improve out-of-domain performance, as, e.g., summarized in a survey by Wang et al. [6]. We focus on the concept of adversarial training. The general idea of adding an adversary to the training of a model was proposed in [7] to improve generative models. The model not only trains on solving the (QA) task, but also tries to compete against an adversary. There are mainly two different types of adversaries that can be incorporated into a training pipeline.
The first adverse model is a discriminator that tries to predict an inputs affiliation to a certain dataset/domain. While the discriminator tries to improve on its classification task, the QA model tries to fool the discriminator while still reducing the QA loss. The intuition is that this approach forces the QA model to generate domain invariant features. If the features are invariant of the specific domain of the data, the model should perform equally well on any domain. This approach is known from training generative adversarial networks (GANs), where the focus lies on producing realistic output to fool a discriminator that tries to distinguish between fake and real images.
The second opponent adds noise to the input embeddings in order to make it harder for the model to reduce the QA loss and prohibit overfitting on brittle feature. The model needs to adapt to the noise and creates representations that are invariant with respect to small perturbations of the input, where the embeddings are perturbed by adding a vector δ from the ϵ-L_∞-ball, i.e. $\|\delta\|_\infty < \epsilon$. This is the classical approach in computer vision tasks to train robust models, which are robust against imperceptible changes to an input image [8, 9]. This concept is very intuitive for images. In NLP applications, the interpretation of perturbed embeddings is less obvious, but as well considered in research, see e.g. [10].

In this project, we analyse those two possible adversaries within a comparison study. We consider two popular approaches to implement first method for adversarial training, and introduce them together with a third approach belonging to the second class of adversaries in section 4. Section 5 contains all details about our comprehensive empirical study and our results, which are further discussed in section 6 and summarized in section 7. We also provide further information on the data and additional experimental results in the appendix.

3 Related Work

In recent years, several apparent limitations of current NLP models have been discussed. For example, Jia et. al. [11] introduced adversarial generated examples, which reduce the average F1 score of successful reading comprehension models trained on the SQuAD [1] dataset from 75% to 36%. Furthermore, Ribeiro et. al. [12] developed a task-agnostic methodology for testing NLP models called CheckList, which also showed the weaknesses in both commercial and state-of-the-art models. Recent research is working on improvements against such failures by, e.g., enhancing domain adaption and domain generalization [6]. There are many ways to improve accuracy of a QA model compared to state-of-the-art approaches, which exclusively consider in-domain data.

Methods that approach out-of-domain (ood) accuracy range from mixture-of-experts methods over data augmentations to meta-learning. Jiang et al. [13] model multiple source domains as a mixture-of-experts and learn a point-to-set metric α to weight the experts for different target examples. Longpre et al. [14] explore data augmentation and sampling techniques for domain-agnostic QA and Li et al. [15] introduced a domain generalization method by dividing the source domain-space into meta-train domains and meta-test domains to simulate train/test domain shift during training by synthesizing virtual testing domains within each mini-batch.

Ganin et al. [16] introduced the idea of domain adaptation through adversarial classification for general backpropagation neural networks. Their approach promotes so called "deep" features that are expressive for the main learning task and also invariant w.r.t. the shift between the domains. As a proof-of-concept, they show promising results on the MNIST digit classification benchmark with domain shifts. Sato et al. [17] then transferred this method to the field of natural language understanding, using adversarial training and a gradient reversal layer (GRL) on a bidirectional LSTM. The model improved their baseline on the task of dependency parsing by up to 6 points on UAS and LAS scores, and we implement this concept as one of our approaches. Lee et al. [18]

made use of the adversarial approach by appending a domain classification discriminator to a pretrained BERT model while fine-tuning on the QA problem. Their results show an improvement of up to 2 points on the F1 score, and it forms the foundation of the second implementation we consider in this project. Liu et al. [10] introduced the ALUM (Adversarial training for large neural LangUage Models) algorithm, which regularizes the training objective by applying perturbations in the embedding space that maximizes the adversarial loss. This is the third approach we include into our empirical study.

An enhancement to the described task of adapting or generalizing beyond in-domain datasets is open-domain QA (OpenQA) [19], which aims to answer a question in the form of natural language based on large-scale unstructured documents. This setting is out of the scope of this report, but we expect a lot of upcoming research to contribute to it.

4 Approach

In this project, we use ideas from GANs [7] and adversarial examples in computer vision [8] to foster domain invariance of QA models. In particular, we incorporate these concepts as described by previous approaches mentioned in section 3, and evaluate their efficiency on improving the QA models out-of-domain (ood) accuracy. We consider 6 datasets, where three are considered as in-domain with a large training corpus and three are considered ood data, since only 127 training examples are available during training for each of them.

The general approach is to use the provided baseline DistilBertForQuestionAnswering [1] and add modifications, such as a domain classifier or a noise vector. The model then still tries to solve the QA task described above, but at the same time competes against an adversarial model that tries to make predictions more difficult. The first adversarial model that we consider is a discriminator trying to correctly predict the domain of the input from the features generated by the DistilBert model. The second model will be a noise vector optimized to make predictions for the QA model harder and is explained at the end of this section.
We start by implementing two variants of a discriminator-based adversarial training approach. In both variants, the training samples are enhanced with a domain label representing the source of the data. The discriminator uses the output vector of the DistilBert model corresponding to [CLS], i.e. the first token in every input sequence, see Figure 1. We implement the discriminator as a MLP with a leaky ReLU activation between each layer and a dropout between the last three layers. More details on the architectures we tested can be found in section 5.3.

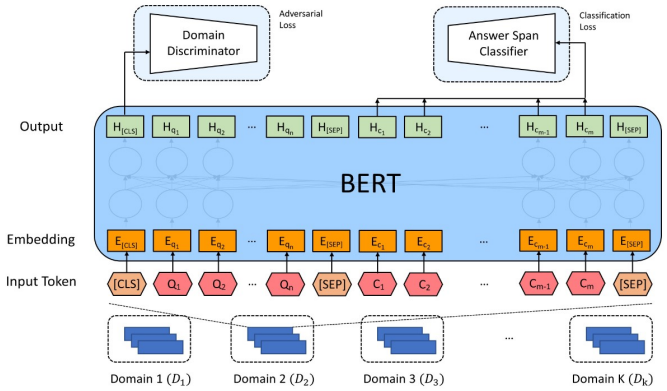

Figure 1: Architecture of ADVA as presented in [18]. It shows a domain discriminator connected to the BERTs hidden output state corresponding to the CLS token.

The approach in [17] employed adversarial training by connecting a domain classifier to the LSTMs output vector via a gradient reversal layer (GRL) and optimize the classifier with a cross entropy loss. The GRL acts as the identity function in the forward step, but multiplies $-\lambda$ to the gradient when computing the gradient. In this way, the part of the model before the GRL, the discriminator, tries to minimize the cross entropy loss, and the part after the GRL, the LSTMs or DistilBERT,

[1] https://huggingface.co/docs/transformers/model_doc/distilbert

tries to maximize it. Finally, the whole model is optimized by minimizing the sum of the cross entropy loss and the task specific loss. We employ this approach by inserting a GRL between DistilBert and the domain classifier. In the remainder of this report, we will refer to this approach as **GRLA**$_{(\text{batch size,epochs},\lambda)}$**D**, where λ is the tuning parameter inside the GRL and D represents the selected discriminator architecture as described in chapter 5.3.

The second approach was presented in [18] and also extends an existing QA model with a discriminator but here the QA model and the domain classifier are trained separately. The [CLS] output vector of DistilBERT is considered as a fixed input when optimizing the discriminator. The loss of the discriminator, l_D, is the cross entropy loss over the different domains. The loss of the QA model is, $l_G = l_{QA} + \lambda l_{D'}$, a weighted sum of the QA loss l_{QA} and the Kullback-Leibler divergence between the logits of the classifiers $\mathbf{x}$ and the uniform distribution, i.e. $l_{D'}(\mathbf{x}) = \sum_i \frac{1}{K}(\log(\frac{1}{K}) - \log(\text{softmax}(x_i)))$. During training of the QA model, the weights of the discriminator are fixed and the loss is averaged across the batch. In the remainder of this report, we will refer to this approach as **ADVA**$_{(\text{batch size,epochs},\lambda)}$**D** with λ being the weight balancing the impact of the discriminator on the loss l_G and D again being the selected discriminator architecture as described in chapter 5.3.

Finally, we come back to the second class of adversarial models, where the adversary is a noise vector added to the embeddings of the input sequence. In particular, we consider the ALUM algorithm suggested in [10]. The algorithm is based on the following so called virtual adversarial training loss,

$$l_{ALUM} = l_{QA}(f(e,\theta), y) + \alpha\, l_{QA}(f(e+\delta_{\max},\theta), f(e,\theta)),$$
$$\text{where } \delta_{\max} = \text{argmax}_{\{\delta:\|\delta\|_\infty<\epsilon\}}\ l_{QA}(f(e+\delta,\theta), f(e,\theta)).$$

In this equation f is our QA model with its parameters θ and the embeddings e as an input. The QA loss l_{QA} measures the difference between the output of the QA model and the true start and end positions of the answer denoted as y. The intuition of the loss function is that we want to improve performance on the QA task, first summand, and minimize the effect of ϵ-perturbations on the model, second summand. The parameter α is responsible for balancing those two objectives. As suggested in [10], we perform one gradient decent step to estimate $\delta_{\max}$. A difference between our ALUM approach and the original article is that we integrate the adversarial training only during fine-tuning and use a standard pretrained model, whereas the initial method was used already during pretraining. We denote our experiments with this method by **ALUM**$_{(\alpha,\epsilon,\text{domain})}$, where domain refers to the data used during training, only in-domain (ind) or also ood data.

5 Experiments

5.1 Data

Our experiments are based on six labeled QA datasets. We consider the three dataset SQuAD[1], NewsQA[20], Natural Questions[21] as in-domain data, since they provide in total 150,000 training examples. The ood datasets are DuoRC[22], RACE[23] and Relation Extraction[24], where for each ood dataset only 127 training samples are available. The detailed distribution over all 6 datasets can be taken from table 4 in the appendix.

Each training sample consists of a context, a question and an answer and we added a domain-label indicating the specific source dataset for each example. The input to the DistilBERT model is constructed by concatenating a [CLS] token, the question, a [SEP] token, the context and another [SEP] token. Since BERT has a maximum context length of 512 restricting the possible input length, long paragraphs will be split and provided to the model in chunks each combined with the corresponding question.

In order to enrich the ood dataset, we used data augmentations to provide more opportunities to learn on ood data but modify them to reduce the overfitting effect. The technique we considered is back translation. The idea is to translate a sentence into another language and back to English using a neural translation pipeline provided by huggingface. In particular, we used the pretrained networks 'Helsinki-NLP/opus-mt-en-de' for German and 'Helsinki-NLP/opus-mt-de-en' for translation back to English. We translated the questions and the context sentences not containing the answer in order for the given answer to remain valid. A few examples can be found in the appendix. Due to the required compute for translation, we only modified the ood datasets and we indicate in our experiments when the data augmentations were used.

5.2 Evaluation method

We evaluate our model by calculating the Exact Match (EM) score and the F1 score. The EM score of an answer given (Question, Context) is either 1, if an exact match in the ground truth answer exist, or 0 if not. The F1 score provides a combination between precision and recall. We take the maximum over all three human-provided answers and the final scores are averaged over the dataset.

To evaluate the accuracy of the implemented domain adaptation methods, we compare its performance on the ood validation set to the baselines performance, which is the standard DistilBertForQuestionAnswering from huggingface trained only on in-domain data. In our experiments, the baseline achieved 48.84 for the F1 and 33.51 for the EM score on the ood dev set.

5.3 Experimental details

For all approaches we start with a learning rate lr = $3e^{-5}$, as proposed in the baseline implementation. We also tried using individual learning rates for both the discriminator and the generator, as introduced in the two time-scale update rule (TTUR) for training GANs by Heusel et al. [25], and we optimized using the AdamW algorithm with $\beta_1 = .5$ and $\beta_2 = .999$.

As described in chapter 4, we use a discriminator to classify the domain-label based on the QA models features corresponding to the [CLS] tokens output vector. In our experiments, we considered different numbers of classes, and started with $c = 6$ classes for all in-domain and ood datasets. In further experiments, the number of classes was reduced to 4 for the 3 in-domain datasets and one additional class for all ood datasets or even to 2 classes for a unified in-domain and a unified ood class. The discriminators architecture has to adapt to the number of classes c, which has to be equal to the size of the output layer. The size of of the first layers varies within our experiments. We started with a basic 3-layer MLP without batchnorm $\mathbf{D}_{(3\text{basic},c)}$ and enhanced it to a small 3-Layer MLP $\mathbf{D}_{(3\text{small},c)}$ with batchnorm which delivered presentable results. The performance for a larger 3-Layer MLP $\mathbf{D}_{(3\text{large},c)}$ even improved, but a further increase to a 4-Layer MLP $\mathbf{D}_{(4,c)}$ does not seem to help. The constant c in the reference for the architectures refers to the number of classes that should be distinguished. The model is enhanced by batchnorm layers and LeakyReLU is used as a non-linear function with a negative slope of .2. For the dropout layers inside the discriminator, we use $p = .2$. A detailed description of the various architectures we used during our study can be seen in the appendix under table 5. For some of the experiments, the discriminator was already pretrained over 3 epochs.

Inspired by Lim et al. [26], we introduce noise into the features **FN** by adding it on the transformers output before feeding it into the qa-layer and the discriminator. This noise follows a Gaussian distribution with mean 0 and standard deviation either $1e-5$ or $\frac{std}{100}$ with std being the standard deviation over all feature values of a batch. If the noise is only used for the answer determination, we will denote it as FN^{QA}. For the in chapter 4 described ALUM method, we again used Gaussian noise with mean 0 and standard deviation $1e-5$, which is added to the embeddings of the input sequence.

For sampling of each batch, we started with the default uniform sampling without replacement as provided within the baseline. In some experiments, we replaced this by a weighted sampling method $\mathbf{WS}_q$ with replacement and q being the probability that a sample is drawn from one of the ood datasets.

The models were mostly trained for 3 epochs, but we also experimented with a smaller and a larger number of epochs, depending on the observed performance at the end of training. We started with a batch size of 16, but promising models were also trained with a batch size of 32.

5.4 Results

5.4.1 ADVA (Adversarial Training Approach)

Since ADVA showed promising results at an early stage of the project, we used many different techniques and applied fine-tuning where applicable, in order to improve the models accuracy on ood data. While the results of the ADVA approach for $\lambda = 1$ already improved the baseline performance, we experienced an additional increase in F1 score by reducing the parameter to

$\lambda = 1e-2$, which was also proposed by Lee et al. [18]. We did not observe any relevant changes when batch size and number of epochs were altered. Reducing the number of classes between which the discriminator should distinguish did not improve the overall performance on ood data. Introducing Gaussian noise to the features shows a promising improvement, if the noise is not too large. A standard deviation of $\frac{std}{100}$ seems to be too large to reliably improve the F1 score, but using a static standard deviation of $1e-5$ consistently shows good results. Unfortunately, weighted sampling only had a small positive impact on the models accuracy on ood data. Instead of learning domain invariant features, the model tends to overfit to the small number of seen ood samples leading to an F1 score on the ood train data of over 80% while improving the baseline performance on the ood validation data by at most 2%. Using two different learning rates resulted in no increase of accuracy.

In our best ADVA experiment w.r.t F1 score, we used the standard setting of $\lambda = 0.01$ and a 3-Layer MLP for the discriminator to distinguish between 6 classes, ADVA$_{(16,3,0.01)}$**D**$_{(3\text{large},6)}$. Figure 4 shows the performance of the best model in pink in comparison to the baseline in orange and the performance of a model with $\lambda = 1$ in blue. Table 7 in the appendix lists all experiments performed for the ADVA method.

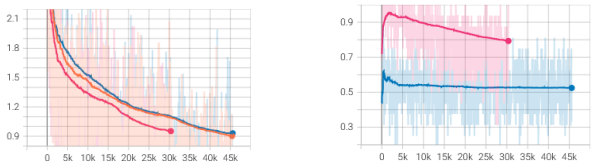

Figure 2: On the left, the training QA-loss is shown, and the right plot shows the discriminators accuracy in predicting the inputs dataset. The orange curve corresponds to the baseline model. The two visualized ADVA experiments differ in $\lambda = 1$ in blue and $\lambda = 0.01$ in pink.

5.4.2 GRLA (Gradient Reversal Layer Approach)

The most crucial tuning parameter in GRLA is the λ inside the gradient reversal layer that decides how strongly the QA model should work against the discriminator. In [17], $\lambda = 0.5$ was proposed. Our experiments indicate that $\lambda = 0.1$ is already too large, since the discriminators accuracy in predicting the domain drops to 50% (guessing) after a few 1000 updates, see blue curve in figure 3. Choosing $\lambda = 0.01$ instead leads to a slower continued decrease of the discriminators accuracy. This indicates that 0.01 is a good level, where the model is able to adapt to the discriminator but it is not too easy. The red curve corresponds to a run using a higher dropout rate on the layer producing the QA-logits to prevent over-fitting. The red curve uses a dropout rate of 0.3, where the other runs use 0.1. The red and pink curves correspond to the best performing runs with GRLA and only slightly differ by favoring different ood datasets. These two experiments are denoted as GRLA$_{(16,3,0.01)}$**D**$_{(3\text{large},6)}$ + Data Augmentation (pink) and GRLA$_{(16,3,0.01)}$**D**$_{(3\text{large},6),p=0.3}$ + Data Augmentation (red), see table 6 in the appendix for F1 and EM scores or section 5.4.4, where the pink run is reported as the best GRLA result.

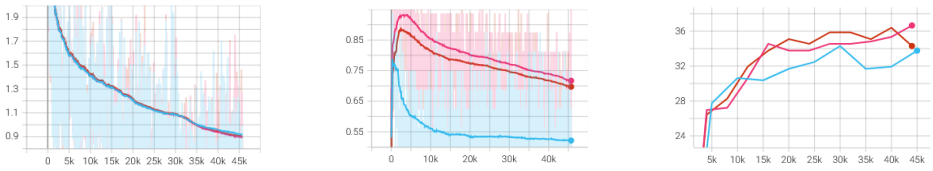

Figure 3: On the left the training loss of the QA task, in the middle the discriminators accuracy in predicting the correct dataset, and on the right the EM score on the ood validation set is display. The GRL experiments differ in $\lambda = 0.1$ in blue and $\lambda = 0.01$ in pink and red.

5.4.3 ALUM (Adversarial training for large neural LangUage Models)

The most influential tuning parameters of the ALUM method are the noise level ϵ and α to weight the importance of the robust adversarial error against the QA loss. The optimal parameter choice

seems to depend on the scale of the QA loss. When running an experiment with $\alpha = 10$ and $\epsilon = 1e-5$ as suggested in [10], the robust error remains constant at around 0.43 over the 3 training epochs and the model barely learns with a loss value of about 4, where standard error and α times the robust error settle, see grey curve in figure 4. When reducing α to 1, the model quickly improves on the QA loss. It is possible that with a larger α the model requires much more training epochs to properly learn the general task. The orange and green curve in figure 4 differ in the use of ood training data, which barely effects the loss functions but improves the performance on the ood validation set. The best performing run used $\alpha = 1$, $\epsilon = 1e-5$ and ood data during training, which corresponds to the green line and is denoted as $\text{ALUM}_{(1,1e-5,\text{ood})}$ in the appendix. The results in terms of F1 and EM score can be found in section 5.4.4 for the best run and in table 8 for all experiments.

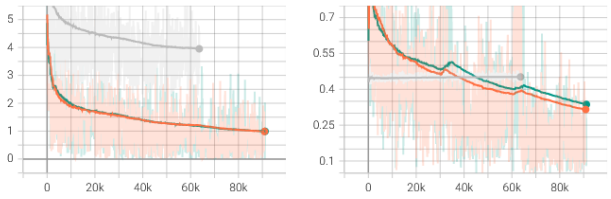

Figure 4: On the left the training loss of the QA task for ALUM with $\alpha = 10$ in grey and $\alpha = 1$ in orange and green. The second graphic shows the robust error, i.e. the QA loss between the model with and without noise added to the embeddings.

5.4.4 Comparison

Both approaches ADVA und GRLA manage to improve the provided baseline performance on the ood dev dataset. ADVA's best F1 score is slightly better than GRLA's best F1 score, although both results are in a similar range and improve the baseline by around 5.8%. Even if ADVA has a slightly higher F1 score, GRLA has a slightly higher quota of exact matches, as given by the EM Score. However, both approaches improve the baseline EM score by around 9%.
On the test set, we achieved the best scores with an ADVA run using a pretrained QA model and discriminator and just using adversarial training for one epoch. It seems that the pretrained QA model was a lucky strike, since later experiments with these hyper-parameters failed [27].

Unfortunately, no improvement in baseline performance in terms of F1 or EM score was evident in our experiments by applying the ALUM approach. Table 1 lists the best results of the three approaches. A comprehensive list of all results can be found in tables 6,7,8 in the appendix.

Method	in-domain dev dataset		out-of-domain dev dataset		
	EM	F1	EM	F1	Improvement in F1
baseline	**54.43**	**70.47**	33.51	48.84	0.0%
GRLA	54.38	**70.47**	**36.65**	51.64	5.73%
ADVA	52.81	68.97	36.39	**51.71**	5.88%
ALUM	50.79	66.59	32.98	47.83	-2.07%

Table 1: Best EM and F1 scores for the three approaches on the in-domain and ood validation dataset. A comprehensive list of all our experiments can be found in the appendix in tables 6,7,8

Method	EM	F1
$\text{GRLA}_{(16,3,0.01)}\mathbf{D}_{(3\text{large},6)}$ + Data Augmentation	40.206	58.582
$\text{ADVA}_{(16,3,0.01)}\mathbf{D}_{(3\text{large},6)}$	40.092	58.471
$\text{ADVA}_{(16,1,0.1)}\mathbf{D}_{(3\text{basic},6)}$ + both pretrained	**41.078**	**59.136**

Table 2: EM and F1 scores on the out-of-domain test set obtained from the test leaderboard.

6 Analysis

It is very surprising to see that both discriminator based adversarial training approaches, GRLA and ADVA, perform almost identically on the test set. Although the two methods rely on slightly different criteria for the model to improve against the discriminator, reducing cross entropy versus reducing Kullback-Leibler divergence with respect to the uniform distribution, the final performance we were able to obtain is comparable and does not clearly favor one approach. However, it is very interesting that GRLA unlike ADVA does not reduce performance on the in-domain validation set compared to the baseline. A next step could be analysing this finding thoroughly. This observation might indicate more room for improvement using GRLA, because the previous advancement on the ood dev set did not happen at the expense of in-domain performance.

One question that comes up when training on only 127 examples for each ood dataset is that the model might already focuses too strongly on the ood train set. Indeed, this might be the case for GRLA and ADVA, where performance is much higher on the ood training set, see table 3 for the corresponding EM and F1 scores. ALUM, even though using ood training data as well, performs only slightly better on the train set compared to the validation set. Our settings of the ALUM algorithm potentially kept the model from learning the QA task deeply.

Besides our model choices on ALUM, a major difference to the original approach [10] is that we could not include ALUM into the pretraining of the DistilBERT model as well. It might require much more training epochs (during pretraining) for the transformer to properly adapt to the noise on the embeddings. Another idea that we had is that ALUM might still improve adversarial robustness of the model, even though it could not increase the ood adaptation. To check this hypothesis, we evaluate our best performing models and the baseline on the adversarial SQuAD dataset[28] provided via the huggingface Dataset package, where we used the 'AddSent' split. However, the results in table 3 do not back our hypothesis, since the performance on adversarial SQuAD seems to change proportionally to the performance on the original SQuAD dataset.

Method	ood validation set		ood train set		(dev) SQuAD		adversarial SQuAD	
	EM	F1	EM	F1	EM	F1	EM	F1
baseline	33.51	48.84	33.51	48.84	62.89	76.91	47.44	59.02
GRLA	36.65	51.64	60.10	74.43	61.82	76.53	46.91	58.91
ADVA	36.39	51.71	56.43	71.73	60.45	75.18	46.26	58.98
ALUM	32.98	47.83	34.91	47.21	59.02	73.52	44.24	56.81

Table 3: EM and F1 scores for the three approaches on the ood-domain validation and train set. The 4 columns on the right contain performance values on the SQuAD and the adversarial SQuAD[28] (AddSent) validation set.

7 Conclusion

In summary, in this project we extended a DistilBERT model by adding an adversarial component with the goal to learn the domain-invariant features for the QA task. There are many ways to introduce such a component and we were able to successfully implement 3 different approaches. Two of these approaches improved the ood accuracy in terms of the F1 score on the ood validation set by about 5%, which shows the general benefit of adversarial training.

Due to the limitations in time and resources, we focused on applying those techniques, that we expected to be most promising, and hence only have a limited number of experiments in which we examined the influence of a single change more closely. In future work, experiments could be conducted that analyse changes of single tuning parameters individually, to examine the impact of each one more thoroughly. When thinking about the next steps, we can imagine to combine adversarial training with additional domain adaptation techniques like a mixture-of-experts or meta learning to improve the QA model. Another open point is why and how GRLA is able to improve on the ood data without a drop in in-domain F1 score, as mentioned in section 6. It could be of general interest to answer this question.

References

[1] Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. Squad: 100,000+ questions for machine comprehension of text. In *CoRR, abs/1606.05250*, 2016.

[2] Minjoon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hannaneh Hajishirzi. Bidirectional attention flow for machine comprehension. *arXiv preprint arXiv:1611.01603*, 2016.

[3] Jacob Devlin Ming-Wei Chang Kenton and Lee Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of NAACL-HLT*, pages 4171–4186, 2019.

[4] Vladimir Karpukhin, Barlas Oğuz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. Dense passage retrieval for open-domain question answering. *arXiv preprint arXiv:2004.04906*, 2020.

[5] 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.

[6] Jindong Wang, Cuiling Lan, Chang Liu, Yidong Ouyang, and Tao Qin. Generalizing to unseen domains: A survey on domain generalization. In *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence*, 2021.

[7] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks, 2014.

[8] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, 2014.

[9] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. *arXiv preprint arXiv:1706.06083*, 2017.

[10] Xiaodong Liu, Hao Cheng, Pengcheng He, Weizhu Chen, Yu Wan, Hoifung Poon, and Jianfeng Gao. Adversarial training for large neural language models. In *arXiv preprint arXiv:2004.08994*, 2020.

[11] Robin Jia and Percy Liang. Adversarial examples for evaluating reading comprehension systems. In *arXiv preprint arXiv:1707.07328*, 2017.

[12] Marco Tulio Ribeiro, Tongshuang Wu, Carlos Guestrin, and Sameer Singh. Beyond accuracy: Behavioral testing of nlp models with checklist. In *In Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, page 4902–4912, 2020.

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

[14] Shayne Longpre, Yi Lu, Zhucheng Tu, and Chris DuBois. An exploration of data augmentation and sampling techniques for domain-agnostic question answering. *In Proceedings of the 2nd Workshop on Machine Reading for Question Answering*, pages 220–227, 2019.

[15] Da Li, Yongxin Yang, Yi-Zhe Song, and Timothy Hospedales. Learning to generalize: Meta-learning for domain generalization. 10 2017.

[16] Yaroslav Ganin and Victor Lempitsky. Unsupervised domain adaptation by backpropagation. In *Proceedings of the 32nd International Conference on Machine Learning*, 2015.

[17] Motoki Sato, Hitoshi Manabe, Hiroshi Noji, and Yuji Matsumoto. Adversarial training for cross-domain universal dependency parsing. In *Proceedings of the CoNLL 2017 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, 2017.

[18] Seanie Lee, Donggyu Kim, and Jangwon Park. Domain-agnostic question-answering with adversarial training, 2019.

[19] Fengbin Zhu, Wenqiang Lei, Chao Wang, Jianming Zheng, Soujanya Poria, and Tat-Seng Chua. Retrieving and reading: A comprehensive survey on open-domain question answering. In *arXiv preprint arXiv:2101.00774*, 2021.

[20] Adam Trischler, Tong Wang, Xingdi Yuan, Justin Harris, Alessandro Sordoni, Philip Bachman, , and Kaheer Suleman. Newsqa: A machine comprehension dataset. In *Association for Computational Linguistics (ACL)*, 2017.

[21] Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Matthew Kelcey, Jacob Devlin, Kenton Lee, Kristina N. Toutanova, Llion Jones, Ming-Wei Chang, Andrew Dai, Jakob Uszkoreit, Quoc Le, and Slav Petrov. Natural questions: a benchmark for question answering research. In *Association for Computational Linguistics (ACL)*, 2019.

[22] Amrita Saha, Rahul Aralikatte, Mitesh M. Khapra, and Karthik Sankaranarayanan. Duorc: Towards complex language understanding with paraphrased reading comprehension. In *Association for Computational Linguistics (ACL)*, 2018.

[23] Guokun Lai, Qizhe Xie, Hanxiao Liu, Yiming Yang, and Eduard Hovy. Ace: Large-scale reading comprehension dataset from examinations. In *EMNLP*, 2017.

[24] Omer Levy, Minjoon Seo, Eunsol Choi, and Luke Zettlemoyer. Zero-shot relation extraction via reading comprehension. In *arXiv preprint arXiv:1706.04115*, 2017.

[25] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessle, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, page 6629–6640, 2017.

[26] Francisco Utrera Winnie Xu Michael W. Mahoney Soon Hoe Lim, N. Benjamin Erichson. Noisy feature mixup. *arXiv:2110.02180*, 2021.

[27] Shiyu Chang Sijia Liu Yang Zhang Zhangyang Wang Michael Carbin Tianlong Chen, Jonathan Frankle. The lottery ticket hypothesis for pre-trained bert networks. In *34th Conference on Neural Information Processing Systems*, 2020.

[28] Robin Jia and Percy Liang. Adversarial examples for evaluating reading comprehension systems. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2021–2031, Copenhagen, Denmark, September 2017. Association for Computational Linguistics.

[29] Adam Fisch, Alon Talmor, Robin Jia, Minjoon Seo, Eunsol Choi, and Danqi Chen. Mrqa 2019 shared task: Evaluating generalization in reading comprehension. pages 1–13, 2019.

A Appendix

This appendix provides additional information on the data and the experiments conducted in this article. For example, in Table 4, further details on the distribution and origin of the used datasets can be found.
We also want to use this additional space to show some examples from our back-translation data augmentation technique. The procedure does not work perfectly on all examples, sometime incorporating mistakes or making to few changes to a sentence. However, the overall impression is satisfying.

original: The lift door closed only after I entered.
back-translated: The elevator door only closed after I entered.

original: Not being fast enough, I was passed by two young people who managed to get into the lift before me.

back-translated: Not fast enough, I got in the elevator in front of me by two young people who made it.

original: What river does Strelna River connect to?
back-translated: Which river connects the Strelna River?

Dataset	**Question Source**	**Passage Source**	**Train**	**Dev**	**Test**
in-domain datasets					
SQuAD [1]	Crowdsourced	Wikipedia	50,000	10,507	-
NewsQA [20]	Crowdsourced	News articles	50,000	4,212	-
Natural Questions [21]	Search logs	Wikipedia	50,000	12,836	-
out-of-domain datasets					
DuoRC [22]	Crowdsourced	Movie reviews	127	126	1248
RACE [23]	Teachers	Examinations	127	128	419
Relation Extraction [24]	Synthetic	Wikipedia	127	128	2693

Table 4: Statistics over in- and out-of-domain Datasets. Both, Question and Passage Source refer to the data collection method for context and answer. The numbers for Train, Dev and Test correspond to the number of data provided within the datasets for each phase. [29]

Discriminator	Architecture
$\mathbf{D}_{(3\text{basic},c)}$	FC_{512}-$LeakyReLU_{0.2}$- FC_{128}-$LeakyReLU_{0.2}$-$Dropout_{0.2}$- FC_{64}-$LeakyReLU_{0.2}$-$Dropout_{0.2}$- FC_c
$\mathbf{D}_{(3\text{small},c)}$	FC_{512}-BN-$LeakyReLU_{0.2}$- FC_{128}-BN-$LeakyReLU_{0.2}$-$Dropout_{0.2}$- FC_{64}-BN-$LeakyReLU_{0.2}$-$Dropout_{0.2}$- FC_c
$\mathbf{D}_{(3\text{large},c)}$	FC_{1024}-BN-$LeakyReLU_{0.2}$- FC_{256}-BN-$LeakyReLU_{0.2}$-$Dropout_{0.2}$- FC_{64}-BN-$LeakyReLU_{0.2}$-$Dropout_{0.2}$- FC_c
$\mathbf{D}_{(4,c)}$	FC_{2048}-BN-$LeakyReLU_{0.2}$- FC_{512}-BN-$LeakyReLU_{0.2}$-$Dropout_{0.2}$- FC_{256}-BN-$LeakyReLU_{0.2}$-$Dropout_{0.2}$- FC_{64}-BN-$LeakyReLU_{0.2}$-$Dropout_{0.2}$- FC_c

Table 5: Detailed description of the architecture of the trained discriminators for domain classification. FC_n refers to a fully connected linear layer with n nodes and BN abbreviates the batchnorm layer. c is the total number of classes that should be distinguished.

Method	out-of-domain dev dataset		
	F1	EM	Improvement in F1
baseline	48.84	33.51	0.0%
$GRLA_{(16,3,0.01)}\mathbf{D}_{(3\text{large},6)}$ + Data Augmentation	51.64	36.65	5.73%
$GRLA_{(16,3,0.01)}\mathbf{D}_{(3\text{large},6),p=0.3}$ + Data Augmentation	50.15	36.39	2.68%
$GRLA_{(16,3,0.01)}\mathbf{D}_{(3\text{large},6)}$	49.27	34.82	0.88%
$GRLA_{(16,3,0.1)}\mathbf{D}_{(3\text{large},6)}$ + Data Augmentation	48.67	34.29	-0.35%
$GRLA_{(16,3,1)}\mathbf{D}_{(3\text{basic},6)}$	48.07	33.77	-1.57%

Table 6: EM and F1 scores for the evaluations on the ood validation dataset of the GRLA experiments

Method	out-of-domain dev dataset		
	F1	EM	Improvement in F1
baseline	48.84	33.51	0.0%
$\text{ADVA}_{(16,3,0.01)}\mathbf{D}_{(3\text{large},6)}$	51.71	36.39	5.88%
$\text{ADVA}_{(16,3,0.01)}\mathbf{D}_{(4,6)}$	51.08	35.60	4.59%
$\text{ADVA}_{(32,3,0.01)}\mathbf{D}_{(3\text{large},6)}$ + $\text{FN}_{(0,1e-5)}$	51.03	35.34	4.48%
$\text{ADVA}_{(16,1,0.1)}\mathbf{D}_{(3\text{basic},6)}$ + both pretrained	50.35	37.17	3.09%
$\text{ADVA}_{(32,3,0.01)}\mathbf{D}_{(3\text{large},6)}$	50.30	35.60	2.99%
$\text{ADVA}_{(16,3,0.01)}\mathbf{D}_{(3\text{small},6)}$	50.11	36.39	2.60%
$\text{ADVA}_{(16,3,1)}\mathbf{D}_{(3\text{basic},6)}$	50.02	35.08	2.41%
$\text{ADVA}_{(32,4,0.01)}\mathbf{D}_{(3\text{small},6)}$ + $\text{WS}_{0.5}$	49.89	34.82	2.15%
$\text{ADVA}_{(16,3,0.01)}\mathbf{D}_{(3\text{large},6)}$ + $\text{FN}^{\text{QA}}_{(0,std/100)}$	49.81	34.55	1.99%
$\text{ADVA}_{(16,3,1)}$ + pretrained $\mathbf{D}_{(3\text{basic},6)}$	49.39	34.03	1.12%
$\text{ADVA}_{(32,2,0.01)}\mathbf{D}_{(3\text{large},6)}$ + $\text{FN}_{(0,1e-5)}$	49.37	35.86	1.09%
$\text{ADVA}_{(32,6,0.01)}\mathbf{D}_{(3\text{small},6)}$ + $\text{WS}_{0.5}$	49.34	34.82	1.02%
$\text{ADVA}_{(16,3,0.01)}\mathbf{D}_{(3\text{large},4)}$ + $\text{WS}_{0.5}$	49.22	35.60	0.77%
$\text{ADVA}_{(32,4,0.01)}\mathbf{D}_{(3\text{large},6)}$	49.20	35.34	0.74%
$\text{ADVA}_{(16,3,0.01)}\mathbf{D}_{(3\text{small},4)}$ + $\text{FN}_{(0,std/100)}$	49.02	32.20	0.37%
$\text{ADVA}_{(16,3,0.01)}\mathbf{D}_{(3\text{small},6)}$	48.92	33.51	0.16%
$\text{ADVA}_{(16,3,0.01)}\mathbf{D}_{(3\text{large},6)}$ + $\text{FN}^{\text{QA}}_{(0,1e-5)}$	48.78	33.77	-0.12%
$\text{ADVA}_{(16,3,0.01)}\mathbf{D}_{(3\text{small},4)}$ + $\text{WS}_{0.5}$	48.74	34.03	-0.2%
$\text{ADVA}_{(32,3,0.01)}\mathbf{D}_{(3\text{small},6)}$ + $\text{WS}_{0.5}$	48.45	34.55	-0.8%
$\text{ADVA}_{(16,3,0.01)}\mathbf{D}_{(3\text{small},2)}$ + $\text{WS}_{0.5}$	48.14	34.03	-1.43%
$\text{ADVA}_{(16,3,0.01)}\mathbf{D}_{(3\text{small},4)}$	47.73	30.89	-2.27%
$\text{ADVA}_{(16,3,0.01)}\mathbf{D}_{(3\text{small},6)}$ + $\text{FN}_{(0,std/100)}$	47.40	30.89	-2.95%
$\text{ADVA}_{(16,3,0.01)}\mathbf{D}_{(3\text{small},4)}$ + $\text{WS}_{0.5}$ + $\text{FN}_{(0,std/100)}$	46.68	33.25	-4.42%
$\text{ADVA}_{(16,3,0.01)}\mathbf{D}_{(3\text{small},6)}$ + $\text{TTUR}_{(3e-5,2e-5)}$	46.50	30.10	-4.79%
$\text{ADVA}_{(16,3,0.01)}\mathbf{D}_{(3\text{small},6)}$ + $\text{TTUR}_{(3e-4,1e-4)}$	42.16	26.18	-13.68%

Table 7: EM and F1 scores for the evaluations on the ood validation dataset of the ADVA experiments

Method	out-of-domain dev dataset		
	F1	EM	Improvement in F1
baseline	48.84	33.51	0.0%
$\text{ALUM}_{(1,1e-5,\text{ood})}$	47.83	32.98	-2.07%
$\text{ALUM}_{(2,1e-6,\text{ind})}$	45.34	29.06	-7.17%
$\text{ALUM}_{(1,1e-5,\text{ind})}$	45.08	29.06	-7.70%
$\text{ALUM}_{(10,1e-5,\text{ind})}$	44.09	29.32	-9.76%

Table 8: EM and F1 scores for the evaluations on the ood validation dataset of the ALUM experiments