A3Net:Adversarial-and-Attention Network for Machine Reading Comprehension

Jiuniu Wang[1,2], Xingyu Fu[1], Guangluan Xu[1(✉)], Yirong Wu[1,2], Ziyan Chen[1], Yang Wei[1], and Li Jin[1]

[1] Key Laboratory of Technology in Geo-spatial Information Processing and Application System, Institute of Electronics, CAS, Beijing, China
gluanxu@mail.ie.ac.cn
[2] School of Electronic, Electrical and Communication Engineering, University of Chinese Academy of Sciences, Beijing, China
wangjiuniu16@mails.ucas.ac.cn

Abstract. In this paper, we introduce Adversarial-and-attention Network (A3Net) for Machine Reading Comprehension. This model extends existing approaches from two perspectives. First, adversarial training is applied to several target variables within the model, rather than only to the inputs or embeddings. We control the norm of adversarial perturbations according to the norm of original target variables, so that we can jointly add perturbations to several target variables during training. As an effective regularization method, adversarial training improves robustness and generalization of our model. Second, we propose a multi-layer attention network utilizing three kinds of high-efficiency attention mechanisms. Multi-layer attention conducts interaction between question and passage within each layer, which contributes to reasonable representation and understanding of the model. Combining these two contributions, we enhance the diversity of dataset and the information extracting ability of the model at the same time. Meanwhile, we construct A3Net for the WebQA dataset. Results show that our model outperforms the state-of-the-art models (improving Fuzzy Score from 73.50% to 77.0%).

Keywords: Machine Reading Comprehension · Adversarial training Multi-layer attention

1 Introduction

Machine reading comprehension (MRC) aims to teach machines to better read and comprehend, and answer questions posed on the passages that they have seen [5]. In this paper, we propose a novel model named Adversarial-and-attention Network (A3Net) for MRC.

M. Zhang et al. (Eds.): NLPCC 2018, LNAI 11108, pp. 64–75, 2018.
https://doi.org/10.1007/978-3-319-99495-6_6

The understanding of neural network is shallow, and it is easy to be disturbed by adversarial examples [7]. So we adopt Adversarial training(AT) [3] as a regularization method to improve our model's generality and robustness. Previous works apply adversarial perturbations mainly on input signals [3] or word embeddings [11], acting as a method to enhance data. While we blend these perturbations into different model layers, especially where question-passage interaction takes place.

The state-of-the-art models have been proved to be effective in English MRC datasets such as CNN/DailyMail [5] and SQuAD [12], such as Match-LSTM [18], BIADF [13], SAN [10], and ReasoNet [14]. They all use attention mechanism and pointer network [17] to predict the span answer. However, these models tend to apply attention function on the limited layer. Thus they would ignore some deep-seated information. To solve this problem, we adopt multi-layer attention to extract information at each level. In the low-level, attention weight is highly affected by the similarity of word embedding and lexical structure(e.g. *affix, part of speech,* etc.), which contains syntactic information. While in the high-level, attention weight reflects the abstract concept correlation between passage and question, which contains semantic information.

To sum up, our contributions can be summarized as follows:

- We blend adversarial training to each layer of our model. Not only can adversarial training enhance the information representation ability, but also improve extraction ability of the whole model.
- We apply multi-layer attention to each layer of our model. In this way, our model can make efficient interactions between questions and passages, so as to find which passage span is needed.
- We propose a novel neural network named A3Net for Machine Reading Comprehension, which gains the best result on the WebQA dataset.

2 Related Work

Adversarial Training. Szegedy et al. [15] found that deep neural network might make mistakes when adding small worst-case perturbations to input. This kind of inputs is called adversarial examples. Many models cannot defend the attack of adversarial examples, including widely used state-of-the-art neural networks such as CNN and RNN. In recent years, there are several methods for regularizing the parameters and features of a deep neural network during training. For example, by randomly dropping units, dropout is widely used as a simple way to prevent neural networks from overfitting.

Adversarial training(AT) [3] is a kind of regularizing learning algorithms. It was first proposed to fine tune the task of image classification. By adding perturbations to input signals during training, neural network gains the ability to tolerant the effect of adversarial example. Miyato et al. [11] first adopt AT to text classification. They add perturbations to word embedding and obtain similar benefits like that in image classification. Following this idea, Wu et al. [20] utilizes AT to Relation Extraction and improves the precision. In order

Table 1. An outline of attention mechanism used in state-of-the-art architectures.

Model	Syntactic attention	Semantic attention	Self-match attention
DrQA [2]	√		
FastQA [19]	√		
Match-LSTM [18]		√	
BIDAF [13]		√	
R-Net [4]		√	√
SAN [10]			√
FusionNet [6]	√	√	√

to improve the model stability and generality, we adopt adversarial training to several target variables within our model.

Attention Mechanism. Attention mechanism has demonstrated success in a wide range of tasks. Bahdanau et al. [1] first propose attention mechanism and apply it to neural machine translation. And then, it is widely used in MRC tasks. Attention near embedding module aims to attend the embedding from the question to the passage [2]. Attention after context encoding extracts the high-level representation in the question to augment the context. Self-match attention [16] takes place before answer module. It dynamically refines the passage representation by looking over the whole passage and aggregating evidence relevant to the current passage word and question.

As shown in Table 1, three different types of attention mechanisms are widely used in state-of-the-art architectures. DrQA [2] simply use a bilinear term to compute the attention weights, so as to get word level question-aware passage representation. FastQA [19] combines feature into the computation of word embedding attention weights. Match-LSTM [18] applies LSTM to gain more context information, and concatenate attentions from two directions. A less memory attention mechanism is introduced in BIDAF [13] to generate bi-directional attention flow. R-Net [4] extends self-match attention to refine information over context. SAN [10] adopts self-match attention and uses stochastic prediction dropout to predict answer during training. Huang et al. [6] summarizes previous research and proposes the fully-aware attention to fuse different representations over the whole model. Different from the above models, we utilize three kinds of method to calculate attention weights, which helps our model to interchange information frequently, so as to select the appropriate words in passages.

3 Proposed Model

In this paper, we use span extraction method to predict the answer to a specific question $Q = \{q_1, q_2, .., q_J\}$ based on the related passage $P = \{p_1, p_2, ..., p_T\}$.

As depicted in Fig. 1, our model can be decomposed into four layers: Embedding Layer, Representation Layer, Understanding Layer and Pointer Layer. And

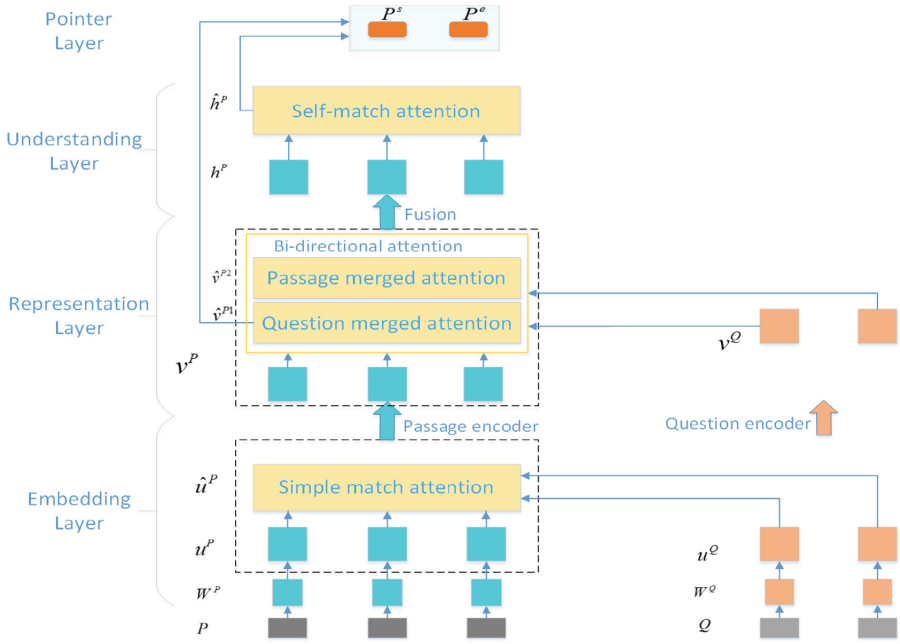


Fig. 1. The overall structure of our model. The dotted box represents concatenate operation.

three attention mechanisms are applied to different layers. Simple match attention is adopted in Embedding Layer, extracting syntactic information between the question and passage. While in Representation Layer, bi-directional attention raises the representation ability by linking and fusing semantic information from the question words and passage words. In Understanding Layer, we adopt self-match attention to refine overall understanding.

3.1 Embedding Layer

Input Vectors. We use randomly initialized character embeddings to represent text. Firstly, each word in P and Q is represented as several character indexes. Afterwards, each character is mapped to a high-density vector space(w^P and w^Q). In order to get a fixed-size vector for each word, 1D max pooling is used to merge character vectors into word vector(u^P and u^Q). Simple match attention is then applied to match word level information, which can be calculated as follows:

$$\hat{u}^P = SimAtt(u^P, u^Q) \tag{1}$$

where $SimAtt(\cdot)$ denotes the function of simple match attention.

Simple Match Attention. Given two sets of vector $v^A = \{v_1^A, v_2^A, ..., v_N^A\}$ and $v^B = \{v_1^B, v_2^B, ..., v_M^B\}$, we can synthesize information in v^B for each vector in v^A. Firstly we get the attention weight α_{ij} between i-th word of A and j-th

word of B by $\alpha_{ij} = softmax(\exp(< v_i^A, v_j^B >))$, where $<>$ represents inner product. Then calculate the sum for every vector in v^B weighted by α_{ij} to get the attention representation $\hat{v}_i^A = \sum_j \alpha_{ij} v_j^B$. This attention variable $\hat{v}^A$ can be denoted as $\hat{v}^A = \{\hat{v}_1^A, \hat{v}_2^A, ..., \hat{v}_N^A\} = SimAtt(v^A, v^B)$.

3.2 Representation Layer

To better extract semantic information, we utilize RNN encoders to produce high-level representation $v_1^Q, ..., v_J^Q$ and $v_1^P, ..., v_T^P$ for all words in the question and passage respectively. The encoders are made up of bi-directional Simple Recurrent Unit (SRU) [8], which can be denoted as follows:

$$v_t^P = BiSRU(v_{t-1}^P, [u_t^P; \hat{u}_t^P]), v_j^Q = BiSRU(v_{j-1}^Q, u_j^Q) \tag{2}$$

Bi-directional attention. is applied in this layer to combine the semantic information between questions and passages. Similar to the attention flow layer in BIDAF, we compute question merged attention $\hat{v}^{P1}$ and passage merged attention $\hat{v}^{P2}$ with bi-directional attention. The similarity matrix is computed by $S_{ij} = \beta(v_i^P, v_j^Q)$, we choose

$$\beta(v_i^P, v_j^Q) = {W_{(S)}}^T [v_i^P; v_j^Q; v_i^P \cdot v_j^Q] \tag{3}$$

where $W_{(S)}$ is trainable parameters, $\cdot$ is element-wise multiplication, [;] is vector concatenation across row.

Question merged attention signifies which question words are most relevant to each passage words. Question merged attention weight (the i-th word in the passage to a certain word in question) is computed by $a_{i:} = \mathrm{softmax}(\mathrm{S}_{i:}) \in \mathrm{R}^J$. Subsequently, each attended question merged vector is denoted as $\hat{v}_i^{P1} = \sum_j a_{ij} v_j^Q$. *Passage merged attention* signifies which context words have the closest similarity to one of the question words and hence critical for answering the question. The attended passage-merged vector is $\tilde{v}^{P2} = \sum_i b_i v_i^P$, where $b = softmax(max_{col}(S))$ and $b \in R^T$, the maximum function $max_{col}()$ performed across the column. Then $\tilde{v}^{P2}$ is tiled T times to $\hat{v}^{P2} \in R^{2d \times T}$, where d is the length of hidden vectors.

3.3 Understanding Layer

The above bi-directional attention representation $\hat{v}_i^{P1}$ and $\hat{v}^{P2}$ is concatenated with word representation v^p to generate the attention representation $\hat{v}^P$.

$$\hat{v}^P = [\hat{v}^{P1}; \hat{v}^{P2}; v^P] \tag{4}$$

Then we use a bi-directional SRU as a Fusion to fuse information, which can be represented as $h_t^P = BiSRU(h_{t-1}^{P-1}, \hat{v}_t^P)$.

In order to take more attention over the whole passage, we apply self-match attention in Understanding Layer. Note that the computing function is the same as simple match attention, but its two inputs are both h^P

$$\hat{h}^P = SimAtt(h^P, h^P) \tag{5}$$

3.4 Pointer Layer

Pointer network is a sequence-to-sequence model proposed by Vinyals et al. [17] In Pointer Layer, we adopt pointer network to calculate the possibility of being the start or end position for every word in the passage. Instead of using a bilinear function, we take a linear function (which is proved to be simple and effective) to get the probability of start position P^s and end position P^e

$$P^s = softmax(W_{Ps}[\hat{h}_i^P; \hat{v}_i^{P1}]) \tag{6}$$

Training. During training, we minimize the cross entropy of the golden span start and end $L(\theta) = \frac{1}{N}\sum_{k}^{N}(\log(P^s_{i^s_k}) + \log(P^e_{i^e_k}))$, where i^s_k, i^e_k are the predicted answer span for the k-th instance.

Prediction. We predict the answer span to be i^s_k, i^e_k with the maximum $P^s_{i^s} + P^e_{i^e}$ under the constraint $0 \leq i^e - i^s \leq 10$.

3.5 Adversarial Training

Adversarial training applies worst case perturbations on target variables. As is shown in Fig. 2, we denote X as the target variable and θ as the parameters of the model. Different from previous works, X can be set as each variable in our model, adversarial training adds adversarial cost function $L_{adv}(X;\theta)$ to the original cost function. The equation of $L_{adv}(X;\theta)$ is described as follows:

$$L_{adv}(X;\theta) = L(X + r_{adv};\theta), r_{adv} = \arg\max_{||r||\leq\varepsilon} L(X + r;\hat{\theta}) \tag{7}$$

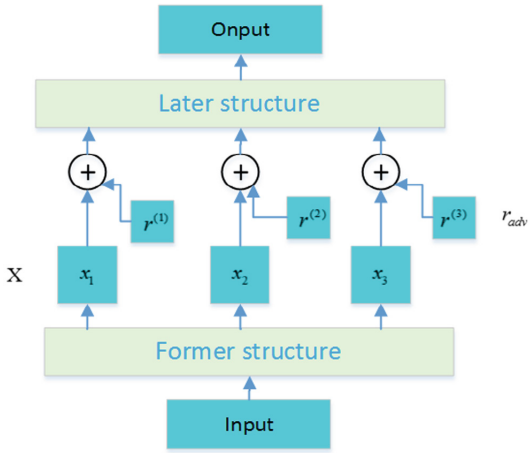


Fig. 2. The computation graph of adversarial training. X denotes target variable, r_{adv} denotes adversarial perturbation. The Input of the model is mapped into target variable X by Former Structure. And then Later Structure generates the Output based on the target variable X combined with adversarial perturbation r_{adv}.

where r is a perturbation on the target variable and $\hat{\theta}$ is a fixed copy to the current parameters. When optimizing parameters, the gradients should not propagate through r_{adv}. One problem is that we cannot get the exact value of r_{adv} simply following Eq. (7), since the computation is intractable. Goodfellow et al. [11] approximate the value of r_{adv} by linearizing $L(X;\hat{\theta})$ near X

$$r_{adv} = \varepsilon||X||\frac{g}{||g||}, g = \nabla_X L(X|\hat{\theta}) \tag{8}$$

where $||\cdot||$ denotes the norm of variable $\cdot$, and ε is an intensity constant to adjust the relative norm between $||r_{adv}||$ and $||X||$. So the norm of r_{adv} is decided by $||X||$, and it could be different during each training sample and training step.

4 Experiments

In this section, we evaluate our model on the WebQA dataset. Outperforming the baseline model in the original paper (Fussy F1 73.50%), we obtain 77.01% with multi-layer attention and adversarial training.

4.1 Dataset and Evaluation Metrics

WebQA [9] is a large scale real-world Chinese QA dataset. Table 2 gives an example from WebQA dataset. Its questions are from user queries in search engines and its passages are from web pages. Different from SQuAD, question-passage pairs in WebQA are matched more weakly. We use annotated evidence(shown in Table 3) to train and evaluate our model. There is an annotated golden answer to each question. So we can measure model performance by comparing predicted answers with golden answers. It can be evaluated by precision (P), recall (Q) and F1-measure (F1):

$$P = \frac{|C|}{|A|}, R = \frac{|C|}{|Q|}, F1 = \frac{2PR}{P+R} \tag{9}$$

Table 2. An example from WebQA.

Question: What kind of color can absorb all the seven colors of the sunshine? (哪种颜色的物体能把太阳的七种颜色全部吸收？)
Passage: Black absorbs light of various colors (黑色能够吸收各种颜色的光)
Answer: Black(黑色)

Table 3. Statistics of WebQA dataset.

Dataset	Question		Annotated evidence	
	#	word#	#	word#
Train	36,145	374,500	140,897	10,757,652
Validation	3,018	36,666	5,412	233,911
Test	3,024	36,815	5,445	234,258

where $|C|$ is the number of correctly answered questions, $|A|$ is the number of produced answers, and $|Q|$ is the number of all questions.

The same answer in WebQA may have different surface forms, such as "Beijing" v.s. "Beijing city". So we use two ways to count correctly answered questions, which are referred to as "strict" and "fuzzy". Strict matching means the predicted answer is identical to the standard answer; Fuzzy matching means the predicted answer is a synonym of the standard answer.

4.2 Model Details

In our model, we use randomly initialized 64-dimensional character embedding and hidden vector length d is set to 100 for all layers. We utilize 4-layer Passage encoder and Question encoder. And Fusion SRU is set to 2-layer. We also apply dropout between layers, with a dropout rate of 0.2. The model is optimized using AdaDelta with a minibatch size of 64 and an initial learning rate of 0.1. Hyper-parameter ε is selected on the WebQA validation dataset.

4.3 Main Results

The evaluation results are shown in Table 4. Different models are evaluated on the WebQA test dataset, including baseline models(LSTM+softmax and LSTM+CRF), BIDAF and A3Net. A3Net(without AT) denotes our base model which does not apply adversarial training(AT); A3Net(random noise) denotes control experiment which replaces adversarial perturbations with random Gaussian noise with a scaled norm. Baseline models utilize sequence label method to mark the answer, while others adopt pointer network to extract the answer. Sequence label method can mark several answers for one question, leading high recall(R) but low precision(P). So we adopt pointer network to generate one answer for each question. In this condition, evaluation metrics(P, R, F1) are equal. Thus we can use *Score* to evaluate our model. Besides, Fuzzy evaluation is closer to real life, so we mainly focus on *Fuzzy Score*.

Based on single layer attention and pointer network, BIDAF obtains the obvious promotion (Fuzzy F1 74.43%). Benefit from multi-layer attention, A3Net

Table 4. Evaluation results on the test dataset of WebQA.

Model	Strict score			Fuzzy Score		
	P(%)	R(%)	F1(%)	P(%)	R(%)	F1(%)
LSTM+softmax	59.38	68.77	63.73	63.58	73.63	68.24
LSTM+CRF	63.72	**76.09**	69.36	67.53	**80.63**	73.50
BIDAF	70.04	70.04	70.04	74.43	74.43	74.43
A3Net(without AT)	71.03	71.03	71.03	75.46	75.46	75.46
A3Net(random noise)	71.28	71.28	71.28	75.89	75.89	75.89
A3Net	**72.51**	72.51	**72.51**	**77.01**	77.01	**77.01**

(without AT) gains 0.97 point promotion in Fuzzy F1 compared to BIDAF, which indicates that multi-layer attention is useful. Our model would get another 1.12 point promotion in Fuzzy F1 after jointly adopting adversarial training on target variable w^P and $\hat{v}^P$.

A common misconception is that perturbation in adversarial training is equivalent to random noise. In actually, noise is a far weaker regularization than adversarial perturbations. An average noise vector is approximately orthogonal to the cost gradient in high dimensional input spaces. While adversarial perturbations are explicitly chosen to consistently increase the cost. To demonstrate the superiority of adversarial training over the addition of noise, we include control experiments which replaced adversarial perturbations with random perturbations from a Gaussian distribution. We use random noise to replace worst case perturbations on each target variable, which only lead slightly improvement. It indicates that AT can actually improve the robustness and generalization of our model.

4.4 Ablation on Base Model Structure

Next, we investigate the ablation study on the structure of our base model. From Table 5 (*A3Net base model* is same with *A3Net (without AT)* in Table 4), we can tell that both Strict Score and Fuzzy Score would drop when we omit any attention. It indicates that each attention layer in A3Net base model is essential.

4.5 Adversarial Training on Different Target Variables

We evaluate the predicted result when we apply adversarial training on different target variables. As is shown in Table 6, applying adversarial training on each target variable can improve Fuzzy Score as well as Strict Score in different degree. It indicates that adversarial training can work as a regularizing method not just for word embeddings, but also for many other variables in our model. Note that the Score is improved significantly when applying AT on embedding variable w^P and attention variable $\hat{v}^P$. It reveals that adversarial training can improve representing ability for both inputs and non-input variables. Finally, we obtain the best result when applying AT on both w^P and $\hat{v}^P$.

Table 5. Comparison of different configurations of base model. The symbols in this table is corresponding with Fig. 1.

Model	Strict score (%)	Fuzzy Score (%)
A3Net base model (without $\hat{u}^P$)	70.57	74.93
A3Net base model (without $\hat{v}^{P1}$)	70.77	75.18
A3Net base model (without $\hat{v}^{P1}$ and $\hat{v}^{P2}$)	70.63	74.56
A3Net base model (without $\hat{h}^P$)	70.70	75.23
A3Net base model	**71.03**	**75.46**

Table 6. Comparison of adversarial training results on different target variables. The symbols in this table is corresponding with Fig. 1

Target variable	Strict score	Fuzzy Score	Target variable	Strict score	Fuzzy Score
None (base model)	71.03	75.46	$\hat{v}^{P1}$	71.85	76.28
w^P	71.95	76.62	$\hat{h}^P$	71.56	76.42
u^P	72.06	76.39	$\hat{v}^P$	72.28	76.81
$\hat{u}^P$	71.32	75.92	w^P and $\hat{v}^P$	**72.51**	**77.01**

We also evaluate adversarial training on two target variables (w^P and $\hat{v}^P$) under different intensity constant ε. As shown in Fig. 3, we repeat experiment 3 times for each target variable on each constant ε, and get the average Fuzzy Score and its std. error. For AT on attention variable $\hat{v}^P$, we obtain the best performance when ε is 0.5×10^{-4}; While for AT on character embedding variable w^P, we obtain the best performance when ε is 2×10^{-4}. It indicates we needs larger adversarial perturbation for low-level variable. While comparable smaller intensity benefits its training for high-level variable. We can explain this phenomenon in two different views. Firstly, w^P and $\hat{v}^P$ are in different concept levels. w^P contains syntactic meaning, and represents as character embedding vectors. Most of the vectors can still hold original meaning under small perturbation, because most points in embedding space have no real meanings. But $\hat{v}^P$ contains semantic meaning. Any perturbation on it would change its meaning, thus our model is sensitive to the perturbation on $\hat{v}^P$. Secondly, w^P and $\hat{v}^P$ are in different layers of our model. $\hat{v}^P$ is closer to Pointer Layer, which could have more influence on the output of the model and computation of cost function.

4.6 Effective of Adversarial Training

Figure 4(a) shows the Fuzzy Score on the test dataset and Fig. 4(b) shows the loss value on the training dataset of A3Net in different configurations. The meaning

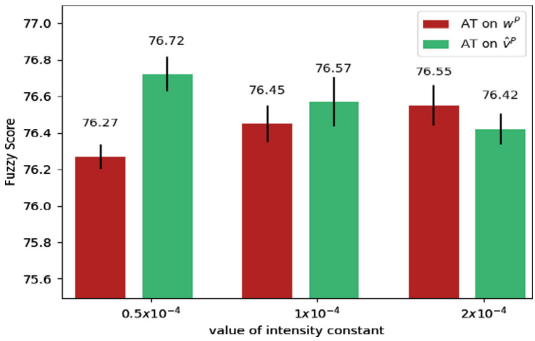
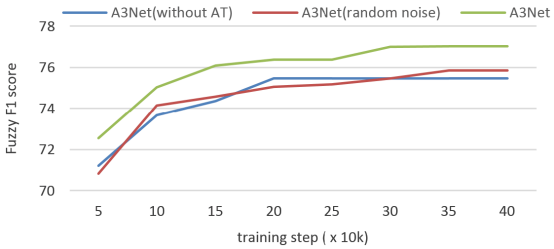
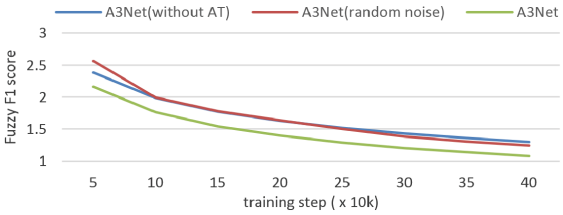


Fig. 3. Effect of intensity constant when AT on target variable w^P and $\hat{v}^P$.



(a) Fuzzy Score (test) under different training step.



(b) Loss value (train) under different training step.

Fig. 4. Fuzzy Score (test) and Loss value (train) under different training step.

of *without AT* and *random noise* are the same with that in Table 4. The data curves of the base model and the random noise model are close to each other in both two figures. It indicates that random noise has limited effect on our model. Within each training step, the Fuzzy Score of our final model is the highest, and its loss value is the lowest. It demonstrates that adversarial training can lead to better performance with less training step.

5 Conclusions

This paper proposes a novel model called Adversarial-and-attention Network (A3Net), which includes adversarial training and multi-layer attention.

Adversarial training works as a regularization method. It can be applied to almost every variable in the model. We blend adversarial training into each layer of the model by controlling the relative intensity of norm between adversarial perturbations and original variables. Results show that applying adversarial perturbations on some high-level variables can lead even better performance than that on input signals. Our model obtains the best performance by jointly applying adversarial training to character embedding and high-level attention representation.

We use simple match attention and bi-directional attention to enhance the interaction between questions and passages. Simple match attention on Embedding Layer refines syntactic information. In addition, bi-directional attention on Representation Layer refines semantic information. Furthermore, self-much

attention is used on Understanding Layer to refine the overall information among the whole passages. Experiments on the WebQA dataset show that our model outperforms the state-of-the-art models.

References

1. Bahdanau, D., Cho, K., Bengio, Y.: Neural machine translation by jointly learning to align and translate. In: Proceedings of ICLR (2015)
2. Chen, D., Fisch, A., Weston, J., Bordes, A.: Reading Wikipedia to answer open-domain questions. In: Proceedings of ACL, pp. 1870–1879 (2017)
3. Goodfellow, I.J., Shlens, J., Szegedy, C.: Explaining and harnessing adversarial examples. In: Proceedings of ICLR (2015)
4. Natural Language Computing Group: R-net: machine reading comprehension with self-matching networks (2017)
5. Hermann, K.M., et al.: Teaching machines to read and comprehend. In: Proceedings of NIPS, pp. 1693–1701 (2015)
6. Huang, H.Y., Zhu, C., Shen, Y., Chen, W.: FusionNet: fusing via fully-aware attention with application to machine comprehension. In: Proceedings of ICLR (2018)
7. Jia, R., Liang, P.: Adversarial examples for evaluating reading comprehension systems. In: Proceedings of EMNLP, pp. 2021–2031 (2017)
8. Lei, T., Zhang, Y.: Training RNNs as fast as CNNs. arXiv preprint arXiv:1709.02755 (2017)
9. Li, P., et al.: Dataset and neural recurrent sequence labeling model for open-domain factoid question answering. arXiv preprint arXiv:1607.06275 (2016)
10. Liu, X., Shen, Y., Duh, K., Gao, J.: Stochastic answer networks for machine reading comprehension. In: Proceedings of NAACL (2018)
11. Miyato, T., Dai, A.M., Goodfellow, I.: Adversarial training methods for semi-supervised text classification. In: Proceedings of ICLR (2017)
12. Rajpurkar, P., Zhang, J., Lopyrev, K., Liang, P.: Squad: 100,000+ questions for machine comprehension of text. In: Proceedings of EMNLP, pp. 2383–2392 (2016)
13. Seo, M., Kembhavi, A., Farhadi, A., Hajishirzi, H.: Bidirectional attention flow for machine comprehension. In: Proceedings of ICLR (2017)
14. Shen, Y., Huang, P.S., Gao, J., Chen, W.: ReasoNet: Learning to stop reading in machine comprehension. In: Proceedings of SIGKDD, pp. 1047–1055. ACM (2017)
15. Szegedy, C., et al.: Intriguing properties of neural networks. In: Proceedings of ICLR (2014)
16. Tan, C., Wei, F., Yang, N., Du, B., Lv, W., Zhou, M.: S-net: from answer extraction to answer generation for machine reading comprehension. In: Proceedings of AAAI (2018)
17. Vinyals, O., Fortunato, M., Jaitly, N.: Pointer networks. In: Proceedings of NIPS, pp. 2692–2700 (2015)
18. Wang, S., Jiang, J.: Machine comprehension using Match-LSTM and answer pointer. In: Proceedings of ICLR (2017)
19. Weissenborn, D., Wiese, G., Seiffe, L.: Making neural QA as simple as possible but not simpler. In: Proceedings of CoNLL, pp. 271–280 (2017)
20. Wu, Y., Bamman, D., Russell, S.: Adversarial training for relation extraction. In: Proceedings of EMNLP, pp. 1778–1783 (2017)