



Training Recurrent Neural Network through Moment Matching for NLP Applications

Yue Deng¹, Yilin Shen¹, KaWai Chen^{1,2}, Hongxia Jin¹

¹AI Center, Samsung Research America, Mountain View, CA, USA

²University of California, San Diego, La Jolla, CA, USA

yuedeng.thu@gmail.com

Abstract

Recurrent neural network (RNN) is conventionally trained in the supervised mode but used in the free-running mode for inferences on testing samples. The supervised mode takes ground truth token values as RNN inputs but the free-running mode can only use self-predicted token values as surrogating inputs. Such inconsistency inevitably results in poor generalizations of RNN on out-of-sample data. We propose a moment matching (MM) training strategy to alleviate such inconsistency by simultaneously taking these two distinct modes and their corresponding dynamics into consideration. Our MM-RNN shows significant performance improvements over existing approaches when tested on practical NLP applications including logic form generation and image captioning.

Index Terms: recurrent neural networks, natural language understanding, moment matching networks

1. Introduction

Recurrent neural networks plays a central role in many speech recognition and natural language processing tasks including acoustic modeling [1], language modeling [2], sequence generation [3], machine translation [4, 5, 6], image captioning [7] and financial applications [8]. RNN models a sequence of discrete data via a joint decomposable distribution over tokens [2, 9]:

$$\begin{aligned} P_{\theta}(\mathcal{W} = \{w_1, w_2 \dots w_T\} | v) \\ = P(w_1 | v) \prod_{t=1}^T P(w_t | w_1 \dots w_{t-1}, v), \end{aligned} \quad (1)$$

where the conditioned variable v serves as a starting token and θ represents the RNN's parameters. In the training phase, all ground truth token values $w_1^{(g)} \dots w_{t-1}^{(g)}$ are observed and can be directly used as RNN's inputs at each time step, i.e. $P(w_t | w_1^{(g)} \dots w_{t-1}^{(g)})$ (see the supervised mode in Fig.1). However, in the inference phase, the RNN can only be implemented in the free-running mode because true token values are no longer available. In such a scenario, the RNN uses its early-step predictions as surrogating inputs to calculate $P(w_t | w_1^{(p)} \dots w_{t-1}^{(p)})$ (see the free-running mode in Fig.1). This discrepancy between supervised and free-running modes may greatly affect the robustness of the trained RNN when it is applied on testing samples.

To alleviate this discrepancy issue, professor forcing (PF) [10] has been proposed from the perspective of adversarial learning [11]. The learning objective of PF is set to fool a discriminator so that it cannot distinguish whether a sequence is generated in the supervised or the free-running mode. Mathematically, this adversarial learning process implicitly imposes a strong regularization over the sequences' probabilistic distributions, which is an effective way to encourage the trained

RNN to also cover the inherent dynamics of the free-running mode. While this distribution matching pursuit behind PF is elegant, its corresponding training procedures are quite challenging. The learning objective of PF is built upon a difficult min-max program requiring very careful optimizations [11]. Such complicated adversarial learning framework does not only increase PF's computational costs but can also result in unstable solution with bad convergence.

The aforementioned discussions naturally lead to a question: is there any alternative method that can achieve the similar goal as PF but exhibits more decent optimization objective? Inspired by the generative moment matching network (GMN) [12], we propose a moment-matching RNN (MM-RNN) training strategy in this work. Our MM-RNN exploits the maximum mean discrepancy (MMD) [12] to explicitly seek for the optimal match of sequences' distributions from either supervised or free-running modes. With appropriate kernel tricks, the MMD can conduct infinite number of statistic comparisons between two distributions [13]. Therefore, it is an ideal surrogate to the complicated adversarial network in distribution matching [12, 14]. More importantly, the objective function of MM-RNN only involves quadratic terms that are highly plausible for optimization purpose.

2. Moment matching for RNN training

In this part, we will introduce the moment matching strategy for RNN training. We will first mathematically define the problem. Then, we detailed how to incorporate the moment matching training strategy and its kernel tricks [15] into RNN for robust and efficient training.

2.1. Preliminaries

We discuss the MM-RNN training by considering (x_i, \mathcal{W}_i^g) as the i th pair of input data and labeled sequence in the training set. There exists a mapping network M that can transform the input data as a hidden representation, i.e. $v_i = M(x_i)$. Here, the input x_i can be data in any type including image, sentence and speech. Then, v_i is used as the initial token to start the whole sequence decoding process in Eq.1. It is possible to run this RNN in either supervised (blue RNN) or free-running (green RNN) mode as shown in Fig.1. In the supervised mode, ground truth token values $w_1^{(g)} \dots w_T^{(g)}$ are directly used as RNN's input at each time step. In the free-running mode, we follow the idea in [10] to sample $w_t^{(p)} \sim P(w_t | w_1^{(p)} \dots w_{t-1}^{(p)}, v_0)$ and feed this sampled token value in RNN at time step t .

The predicted sequence obtained in supervised (resp. free-running) mode is denoted as $\mathcal{W}_i^s = \{w_{i1}^s \dots w_{iL}^s\}$ (resp. $\mathcal{W}_i^f = \{w_{i1}^f \dots w_{iL}^f\}$). For the ease of comparisons at sequence level, we further design an encoding network E to map the generated

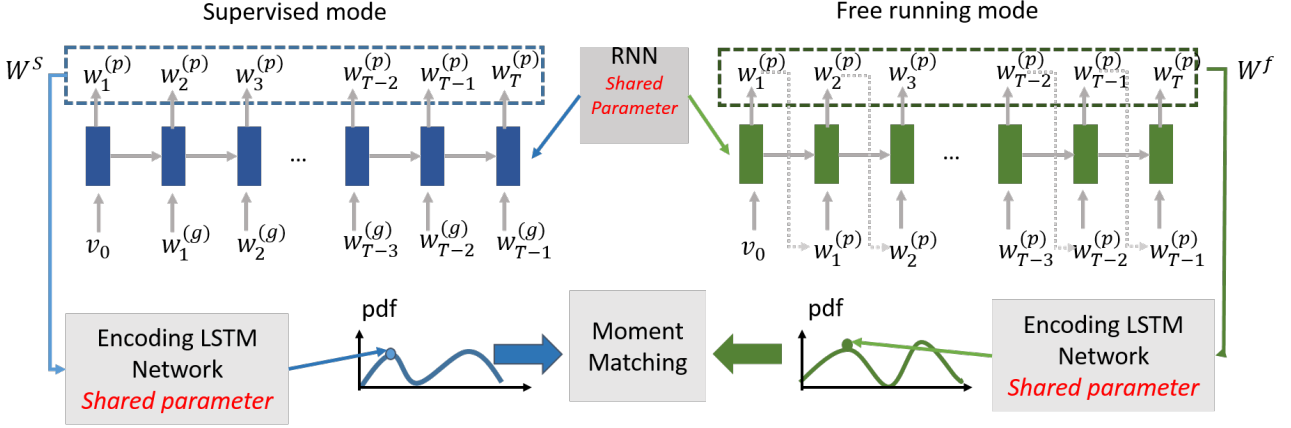


Figure 1: An overview of moment matching recurrent neural networks training.

sentences \mathcal{W}_i^s (resp. \mathcal{W}_i^f) into a hidden vector $h_i^s = E(\mathcal{W}_i^s)$ (resp. $h_i^f = E(\mathcal{W}_i^f)$). There are many different ways to design this sequence encoder and we just consider the most widely used bi-directional LSTM implementation [16]. In Fig. 1, we noted that the two encoding LSTM share the same parameter meaning they are essentially the same. The matching pursuit is to minimize the discrepancy between generative probabilities of $h^f = \{h_i^f\}_{i=1}^N$ and $h^s = \{h_i^s\}_{i=1}^N$. In an ideal case, if there is no difference between these two distributions, the corresponding RNN dynamics in supervised and free-running mode should also have no distinction.

To achieve this goal, professor forcing (PF) has introduced an auxiliary discriminator neural network to perform adversarial learning. As indicated in the benchmark GAN works [11, 17], the discriminator neural network exactly serves the purpose for distribution comparisons. However, the training process of the professor forcing model is not easy. Therefore, in this work, we consider a more efficient and intuitive approach to non-parametrically compare two pdfs accumulated from h^f and h^s .

2.2. Moment matching

We utilize a technique from statistical hypothesis testing known as maximum mean discrepancy (MMD) to compare statistics between samples of these two distributions. We follow the idea in [12] and adopt the mean squared differences as the statistics of sets h^f and h^s :

$$\begin{aligned}
 L_{MMD}(h^s, h^f) &= \frac{1}{N^2} \left\| \sum_{i=1}^N \psi(h_i^s) - \sum_{i=1}^N \psi(h_i^f) \right\|^2 \\
 &= \frac{1}{N^2} \left[\sum_{i=1}^N \sum_{i'=1}^N \psi(h_i^f)^T \psi(h_{i'}^f) \right. \\
 &\quad \left. - 2 \sum_{i=1}^N \sum_{j=1}^N \psi(h_i^f)^T \psi(h_j^s) + \sum_{j=1}^N \sum_{j'=1}^N \psi(h_j^s)^T \psi(h_{j'}^s) \right]
 \end{aligned} \quad (2)$$

where $\psi(\cdot)$ is considered as a general mapping function. If it is fixed as the identity function, the loss in Eq.2 just simply compares the sample mean of two distributions. Alternative choices of ψ can enable the match of higher order moments.

We noted that terms in Eq.2 only involve inner products of ψ vectors and, therefore, the kernel trick can be applied. We replace all inner product terms in Eq.2 with a kernel function $k(\cdot, \cdot)$, i.e. $\psi(y_i)^T \psi(y_j) = k(y_i, y_j)$. The kernel trick exhibits

two indispensable properties for distribution matching. First, kernel maps the sample vector into an infinite dimensional feature space. In the universal reproducing kernel Hilbert space, $L_{MMD}(h^s, h^f) = 0$ if and only if $P(h^s) = P(h^f)$ [18]. Secondly, when we choose universal kernels like Gaussian kernel as the detailed kernel function $k(\cdot, \cdot)$, it is easy to get an Taylor expansion of the feature map ψ that contains infinite number of terms covering all orders of statistics. This trick allows the matching of all moments of two statistics [12]. In this paper, we choose the Gaussian kernel as the default kernel function in the L_{MMD} computation. This particular selection makes the final L_{MMD} objective only involve quadratic terms, which are easily optimized in the RNN's back-propagation learning step.

We can now define the whole objective function for MM-RNN:

$$\begin{aligned}
 \min_{(\theta_M, \theta_E, \theta)} \mathcal{L} &= L_C(\mathcal{W}^s, \mathcal{W}^g) + L_C(\mathcal{W}^f, \mathcal{W}^g) \\
 &\quad + \lambda L_{MMD}[E(\mathcal{W}^s), E(\mathcal{W}^f)] \\
 \text{s.t. } \mathcal{W}^s &= \{\mathcal{W}_i^s\}_{i=1}^N, \mathcal{W}_i^s \sim P_\theta^s(\mathcal{W} | v_i = M(x_i)) \\
 \mathcal{W}^f &= \{\mathcal{W}_i^f\}_{i=1}^N, \mathcal{W}_i^f \sim P_\theta^f(\mathcal{W} | v_i = M(x_i))
 \end{aligned} \quad (3)$$

In Eq.3, \mathcal{W}^s (resp. \mathcal{W}^f) are sampled sequences from $P_\theta^s(\mathcal{W})$ (resp. $P_\theta^f(\mathcal{W})$) in the supervised (resp. free-running) mode. We noted that $P_\theta^s(\mathcal{W})$ and $P_\theta^f(\mathcal{W})$ shares the same network parameter θ implying they are in fact defined from the same RNN. But this same RNN can be run in different modes. The term L_C denotes the cross entropy loss that measures the difference between generated sequence and ground truth [19]. L_{MMD} has been discussed in Eq.2 that tries to match the sequences sampled in different RNN modes.

In Eq.3, in addition to the basic RNN (paired with parameter θ), there are also two extra neural networks M and E with parameter sets θ_M and θ_E to be learned through optimization. M is the feature mapping network that transforms the input data as a initial token for RNN. This mapping network's structure can only be determined when coming to a particular task [20]. E is the encoder neural network that transforms the generated sequences as hidden vectors for moment matching. We follow the same idea in professor forcing [12] to implement this encoder network by a bi-directional LSTM. All these three networks can be learned in an end-to-end manner by the ADAM optimizer [21]. The detailed training steps for MM-RNN have been provided in Algorithm 1.

Algorithm 1: MM-RNN training

Input : A training dataset containing pairs of input data and its labeled sequences.

Initialization: Initialize θ_M in mapping network M , θ_E in encoder network E and θ in RNN

- 1 **for** $k=1 \dots K$ **do**
- 2 Sample a minibatch of N samples X and their labeled sequences \mathcal{W}^g from training set;
- 3 Get $v_i = M(x_i), i = 1 \dots N$;
- 4 Use v_i and θ to run RNN in supervised mode and get its corresponding output sequence $\mathcal{W}^s = \{\mathcal{W}_i^s\}_{i=1}^N$;
- 5 Use v_i and θ to run RNN in free-running mode and get its corresponding output $\mathcal{W}^f = \{\mathcal{W}_i^f\}_{i=1}^N$;
- 6 Take $\mathcal{W}^g, \mathcal{W}^s$ and \mathcal{W}^f in Eq.(3) for loss calculation and back-propagate the loss to update networks' parameters θ_M, θ_E and θ ;
- 7 **end**

Output : The well trained RNN with parameter θ ;

3. Experiments

In this part, we will conduct experiments on two natural language processing (NLP) tasks including logic form generation and image captioning. Both of these works rely on an explicit RNN structure for sequence generation. The MM-RNN training will be compared with the original RNN training strategy (Ori.) and professor forcing (PF).

3.1. Logic Form Generation from Utterance

Table 1: Examples of two logic form generation datasets with argument identification

Dataset	Length	Examples
ATIS	9.8	nonstop flight ci1 to ci0
	22.9	(lambda \$0 e (and(flight \$0)(nonstop \$0) (from \$0 ci1)(to \$0 ci0)))
GEO	7.6	what state capital is c0
	19.1	(lambda \$0 e (and(state:t \$0)(capital:t \$0 c0)))

In this part, we applied MM-RNN for logic form generation from human utterances which is a fundamental problem in natural language understanding. This is because computers can only understand commands in the format of logic forms rather than our human natural languages. For instance, the generated logic form can be used as the query to a knowledge base for questioning and answering. Unfortunately, human beings do not express their commands in the way like logic forms. Therefore, it is a critical step in NLU to map natural language utterances into executable logic forms.

The logic form generation task can be tackled as a seq2seq learning problem in which the input sequence is an utterance and the output is the desired logic form. Here, we conduct experiments on two widely used logic form datasets including ATIS (5410 queries to a flight booking system) and GEO (880 queries about U.S. geography). Some examples of utterances with their annotated logic forms are provided in Table 1. The average lengths for input and output sequences in these two datasets are also reported. To improve the robustness in handling unknown new words, we follow the augmentation identi-

fication approach [22] to replace entities and numbers in input with their underlying type names and unique IDs. For instance, in Table 1, we replace exact city names with symbols 'c0' and 'c1' on the shown sentence in ATIS dataset. To conduct seq2seq task, words of input sequence and symbols of the output sequence are both converted as a series of one-hot vectors with one 'hot' entry indicating a certain word/symbol. Specifically, we tried both the LSTM and the attention model on the decoder side. For the attention model, we use the same structure as introduced in [22]. In this work, all LSTM are implemented with 128 hidden states.

Table 2: Accuracy of the logic form generation task %

Feature encoder		ATIS	GEO
LSTM	Original	81.6±1.1	80.2±0.8
	PF	82.5±0.7	82.9±0.7
	MM	84.2±0.7	84.4±0.8
Attention	Original	82.0±1.1	81.7±0.8
	PF	83.8±0.9	83.2±0.9
	MM	85.7±0.9	85.3±0.8

In each dataset, 80% sentences are randomly selected for training and the remaining 20% sentences are uniformly divided into validation and testing sets. We repeat such random splitting processes for 10 times. The results of three training strategies are reported in Table 2. In the table, PF and MM-RNN generally improves the original RNN model on two datasets. The MM-RNN further improves the performances of PF by using any feature extractor. The best results on these two datasets are obtained by our MM-RNN with Attention-based feature extractor. These experimental findings suggest the equal importance of network structure and training methods for the final performance.

We also report the results by conducting MM-RNN on the provided training and testing data splits in [22]. MM-RNN obtains the accuracy of 87.1% and 86.8% on ATIS and GEO, respectively. These results are 2.6% and 2.0% higher than the reported accuracy in the original paper. Computational costs on the larger ATIS dataset are also recorded. Original RNN takes 674 seconds to finish 100 epochs. MM-RNN and PF respectively spends 933 seconds and 1,436 seconds on the same task. From such comparisons, we get to the conclusion that MM-RNN only slightly increases the training complexity but significantly improves the model performances than the original RNN model. It also beats the prevalent PF model in both speed and accuracy.

3.2. Image captioning

In the second task, we applied MM-RNN to a multimodal application for image captioning. We train and evaluate our model using the commonly used MSCOCO dataset [23], which consists of 82,783 images for training, 40,504 for validation, and 40,775 for testing. We noticed that each image corresponds to 5 ground truth captions. We follow Karpathy et al.[24] to preprocess the sentences, where all the words are converted to lower-case, and all non-alphanumeric characters are discarded. We use a word threshold of 2 (*i.e.* discard all words appear less than twice) to remove rare words through all sentences. For result evaluation, we use the coco-caption evaluation API to calculate BLEU 1~4 and METEOR (MET) as the reported accuracy in Table 3.



RNN.: a man on a surfboard riding a wave

PF: a man riding a wave on top of a surfboard

MM: a man riding a surfboard on a wave in the ocean



RNN.: a group of people flying kites in a field

PF: a group of people flying kites in a field

MM: a crowd of people flying kites over a sandy beach



RNN.: a close up of a plate of food on a table

PF: a plate of chicken, broccoli, asparagus, and cauliflower

MM: a close up of a plate of food with meat and broccoli



RNN.: a flock of birds flying over a sandy beach

PF: a group of people riding horses along a sandy beach

MM: a flock of birds standing on top of a sandy beach

Figure 2: Image captioning results by training the captioning RNN with original (black), professor forcing (blue) and moment matching (red). It is observed from the comparisons that PF can cover more aspects of the explained image and MM intends to over-explain the image with non-appearing scenarios.

The decoding RNN is trained by original method, professor forcing and MM-RNN. Some image captioning results by these three methods are shown in Fig.2 with attention model as the image feature extractor. From these empirical results, we have found that PF and MM can generate sentences covering more details about the input image than the original method. The performances of MM-RNN are even better than PF when comparing the corresponding results in the third *broccoli* and fourth *bird* images in Fig.2. PF generates wrong sentences that have described items not covered in the image. However, MM-RNN are quite reliable that just summarized appearing scenarios without over explanation. These shown cases intuitively explains the advantages of PF over others. In addition, we will provide more quantitative evaluations to compare different approaches.

Table 3: Image captioning results on MSCOCO by using both CNN and attention model as the basic visual feature extractor. The sentence decoder structure is the same but are trained with different methods.

	B1	B2	B3	B4	MET
CNN+Ori.	0.666	0.451	0.304	0.203	0.192
CNN+PF	0.723	0.471	0.335	0.251	0.223
CNN+MM	0.741	0.544	0.382	0.281	0.253
Atten.+Ori.	0.718	0.504	0.357	0.250	0.230
Atten.+PF	0.723	0.512	0.367	0.261	0.234
Atten.+MM	0.775	0.563	0.417	0.302	0.257

We further report BLEU 1~4 and METEOR (MET) as quantitative measures to compare these three RNN training strategies in Table 3. MM-RNN beats all other two methods on this image captioning task. We also reported the computational costs of different training approaches. The complexity are reported in Table.4 by using either convolution neural network (CNN) or attention model as the image feature extractor. The decoder part are implemented by the same LSTM structure but are trained via three different methods including original training, professor forcing and our moment matching strategy. All models are implemented by TensorFlow with 16 GPUs and the time costs are reported in hours. From the computational costs comparisons, we have observed that MM-RNN is a bit heavier than the original RNN training but is far more efficient

than professor forcing. This is because both RNN and MM shows an explicit objective function that can be optimized directly. However, the PF involves the min-max optimizations of multiple neural networks which is more complicated than ours.

Table 4: Computational time comparisons by using different image feature extractors and different RNN training methods (in hours)

	CNN Extractor	Attention Model
RNN.	1.5h	2.7 h
PF	2.6h	4.9h
MM	1.8h	3.4h

4. Conclusions

We have introduced a MM-RNN training strategy to fill the gap of implementing RNN in the training and inference phases. MM-RNN is simple in concept, fast in implementation and reliable in performance. We ascribe the successes of it to the inspiring moment matching learning objective in Eq.2. On the other hand, while its competitor professor forcing also addressed the similar learning purpose, its optimization yield to a more complicated framework with a min-max adversarial loss. Therefore, it is less efficient and less robust than the MM-RNN model. In this task, we mainly consider experiments on the machine translation (logic form generation) and image captioning tasks because these two tasks are widely regarded as the most challenging machine learning topics in the contemporary AI research. RNN training is a common topic that has encouraged a large number of real world applications in the speech, natural language and finance field. We hence believe more complicated and practical problems can be well solved by this general moment matching strategy in the future.

5. References

- [1] H. Sak, A. Senior, and F. Beaufays, "Long short-term memory recurrent neural network architectures for large scale acoustic modeling," in *Fifteenth annual conference of the international speech communication association*, 2014.
- [2] T. Mikolov, M. Karafiát, L. Burget, J. Cernocký, and S. Khudanpur, "Recurrent neural network based language model." in *Inter-speech*, vol. 2, 2010, p. 3.

- [3] L. Yu, W. Zhang, J. Wang, and Y. Yu, "Seqgan: Sequence generative adversarial nets with policy gradient." in *AAAI*, 2017, pp. 2852–2858.
- [4] K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using rnn encoder-decoder for statistical machine translation," *arXiv preprint arXiv:1406.1078*, 2014.
- [5] M.-T. Luong, H. Pham, and C. D. Manning, "Effective approaches to attention-based neural machine translation," *EMNLP*, 2015.
- [6] N. Kalchbrenner and P. Blunsom, "Recurrent continuous translation models." in *EMNLP*, vol. 3, no. 39, 2013, p. 413.
- [7] O. Vinyals, A. Toshev, S. Bengio, and D. Erhan, "Show and tell: A neural image caption generator," in *CVPR*, 2015, pp. 3156–3164.
- [8] Y. Deng, F. Bao, Y. Kong, Z. Ren, and Q. Dai, "Deep direct reinforcement learning for financial signal representation and trading," *IEEE transactions on neural networks and learning systems*, vol. 28, no. 3, pp. 653–664, 2017.
- [9] F. Bao, Y. Deng, M. Du, Z. Ren, Q. Zhang, Y. Zhao, J. Suo, Z. Zhang, M. Wang, and Q. Dai, "Probabilistic natural mapping of gene-level tests for genome-wide association studies," *Briefings in bioinformatics*, p. bbx002, 2017.
- [10] A. M. Lamb, A. G. A. P. GOYAL, Y. Zhang, S. Zhang, A. C. Courville, and Y. Bengio, "Professor forcing: A new algorithm for training recurrent networks," in *Advances In Neural Information Processing Systems*, 2016, pp. 4601–4609.
- [11] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *NIPS*, 2014, pp. 2672–2680.
- [12] Y. Li, K. Swersky, and R. Zemel, "Generative moment matching networks," in *Proceedings of the 32nd International Conference on Machine Learning (ICML-15)*, 2015, pp. 1718–1727.
- [13] A. Gretton, K. M. Borgwardt, M. Rasch, B. Schölkopf, and A. J. Smola, "A kernel method for the two-sample-problem," in *Advances in neural information processing systems*, 2007, pp. 513–520.
- [14] D. Ulyanov, V. Lebedev, A. Vedaldi, and V. S. Lempitsky, "Texture networks: Feed-forward synthesis of textures and stylized images." in *ICML*, 2016, pp. 1349–1357.
- [15] Y. Deng, Y. Zhao, Z. Ren, Y. Kong, F. Bao, and Q. Dai, "Discriminant kernel assignment for image coding," *IEEE transactions on cybernetics*, vol. 47, no. 6, pp. 1434–1445, 2017.
- [16] A. Graves and J. Schmidhuber, "Framewise phoneme classification with bidirectional lstm and other neural network architectures," *Neural Networks*, vol. 18, no. 5-6, pp. 602–610, 2005.
- [17] Y. Deng, Y. Shen, and H. Jin, "Disguise adversarial networks for click-through rate prediction," in *Proceedings of the 26th International Joint Conference on Artificial Intelligence*. AAAI Press, 2017, pp. 1589–1595.
- [18] A. Gretton, K. M. Borgwardt, M. J. Rasch, B. Schölkopf, and A. Smola, "A kernel two-sample test," *Journal of Machine Learning Research*, vol. 13, no. Mar, pp. 723–773, 2012.
- [19] Y. Deng, F. Bao, X. Deng, R. Wang, Y. Kong, and Q. Dai, "Deep and structured robust information theoretic learning for image analysis," *IEEE Transactions on Image Processing*, vol. 25, no. 9, pp. 4209–4221, 2016.
- [20] Y. Deng, Z. Ren, Y. Kong, F. Bao, and Q. Dai, "A hierarchical fused fuzzy deep neural network for data classification," *IEEE Transactions on Fuzzy Systems*, vol. 25, no. 4, pp. 1006–1012, 2017.
- [21] D. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv:1412.6980*, 2014.
- [22] L. Dong and M. Lapata, "Language to logical form with neural attention," *ACL*, 2016.
- [23] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft coco: Common objects in context," in *ECCV*. Springer, 2014, pp. 740–755.
- [24] A. Karpathy and L. Fei-Fei, "Deep visual-semantic alignments for generating image descriptions," in *CVPR*, 2015, pp. 3128–3137.