# Learning Assistance from an Adversarial Critic for Multi-Outputs Prediction

**Yue Deng, Yilin Shen** and **Hongxia Jin**
AI Research Center, Samsung Research America
{y1.deng, yilin.shen, hongxia.jin}@samsung.com

## Abstract

We introduce an adversarial-critic-and-assistant (ACA) learning framework to improve the performance of existing supervised learning with multiple outputs. The core contribution of our ACA is the innovation of two novel modules, *i.e.* an 'adversarial critic' and a 'collaborative assistant', that are jointly designed to provide augmenting information for facilitating general learning tasks. Our approach is not intended to be regarded as an emerging competitor for tons of well-established algorithms in the field. In fact, most existing approaches, while implemented with different learning objectives, can all be adopted as building blocks seamlessly integrated in the ACA framework to accomplish various real-world tasks. We show the performance and generalization ability of ACA on diverse learning tasks including multi-label classification, attributes prediction and sequence-to-sequence generation.

## 1 Introduction

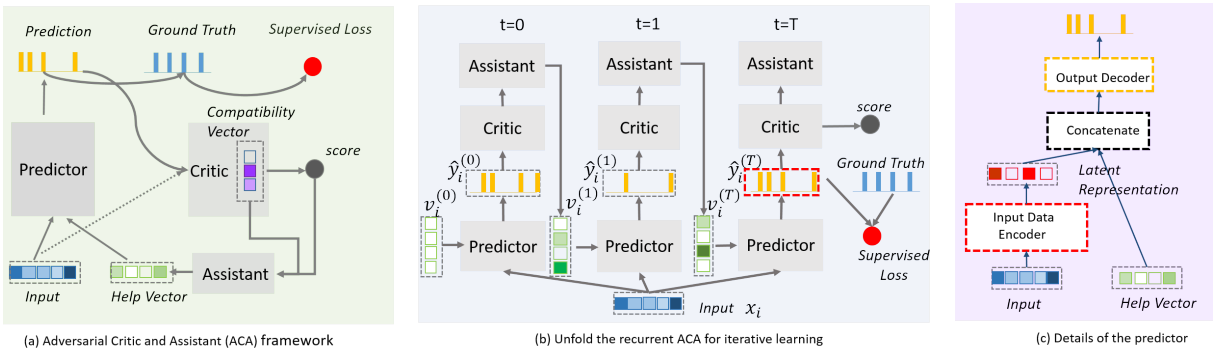
Learning a predictor $f(\cdot)$ that could map raw input data $x$ as meaningful output $y$ is a fundamental pursuit across many domains. In the conventional setting, learning algorithms mainly consider single-output scenarios, where $y$ is either regarded as a one-hot vector in classification or a single value in regression. However, with the progresses of deep learning, predicting multiple outputs from the raw data has been becoming a highly desired research topic in many practical problems[Lin *et al.*, 2014; Vinyals *et al.*, 2015]. Such multi-outputs can be the topic distribution in a document, multiple attributes of an image or even a sentence composed of multiple words.

We introduced a new learning framework named adversarial-critic-and-assistant (ACA) for multiple-outputs predictions. Our ACA underscores the nature of the studied problem by augmenting traditional "predictor" with two novel modules, *i.e.* a 'critic' and an 'assistant'. The critic module was inspired by the breakthrough adversarial learning [Goodfellow *et al.*, 2014] that conducts adversarial gambling with the predictor to evaluate the quality of the generated outputs. For instance, in the machine translation task, the critic module could evaluate whether the generated sentence is fluent enough as human language. In image captioning task, it evaluates whether two attributes are likely to coherently appear in the same scenario. Unlike existing critic modules that just adopted a single score for evaluating, we believe there should be very rich information inherently encoded in the critic. Therefore, we designed an assistant module to extract such inherent information as useful feedbacks to further improve the predictor. In a nutshell, the critic evaluates "how" good the predictor is and the assistant module further illustrates "why"the critic reaches such a conclusion. These two parts offer auxiliary supervised information to help the learning system generate better result with more reasonable multi-output structure.

ACA does not play the role of a competitor but is more preferable to be regarded as a complementary framework that could potentially improve many existing algorithms' performances. In this paper, we will show the compatibility of ACA on two challenging machine learning tasks including multi-label classification and sequence-to-sequence learning. These models are further applied to solve a number of real-world tasks such as documents modeling, image attributes prediction and logic form generation for natural language understanding (NLU). Among these three tasks, the only variation on ACA just appears on the design of the predictor module. From the experimental results, it will be shown that the assistant module is able to improve many algorithms' performances on various tasks.

## 2 ACA Framework

The intuition of ACA learning could be conceptually comprehended by imaging the interaction among a student, a professor and a teaching assistant (TA) in a quiz process. Student takes a quize, professor evaluates the results and more importantly, the TA further summarizes the professor's comments to the student. In our ACA framework, these three roles were respectively represented by: predictor (student), critic (professor) and assistant (TA) as shown in Fig 1 (a). There are two types




(a) Adversarial Critic and Assistant (ACA) framework (b) Unfold the recurrent ACA for iterative learning (c) Details of the predictor

Figure 1: (a) An overview of Adversarial Critic and Assistant Framework. The solid lines indicate the main interactions among these three modules. The dotted line is added to indicate that the critic also takes the original input for compatibility scoring. (b) Unfolding the recurrent ACA framework into an iterative learning process involving multiple steps $T$. $T = 2$ in this figure. The dotted link in (a) is omitted here for the sake of simple representation. (c) The detailed configuration of the 'predictor' function.

of interactions among these three roles. The first interaction is the collaboration between the student and the TA. Their purposes are quite well aligned to ultimately improve student's learning ability so that he can get a high quiz score. The second type of interaction is an adversarial gambling between the student and professor as discussed in [Deng *et al.*, 2019].

Unlike previous work [Deng *et al.*, 2019], the ACA framework mainly contributes in bringing the concepts of the 'Assitant' module and its inherent feedback meachnism into consideration. Mathematically, we use $f_P(\cdot)$, $f_C(\cdot)$ and $f_A(\cdot)$ to respectively define the transformations of the 'Predictor', the 'Critic' and the 'Assistant' in Fig.1(a) which can all be implemented with neural networks. The predictor gets a raw input $x_i$ and predicts a multi-output vector $\hat{y}_i$, which are fed into the critic module for compatibility scoring altogether with input data $x_i$ ,

$$s_i = f_C(x_i, \hat{y}_i) = sigmoid(g(h_{(x_i,\hat{y}_i)})) \quad (1)$$

where $s_i$ is the score measuring the compatibility of the input data $x_i$ and the predicted multi-output vector $\hat{y}_i$ [Gygli *et al.*, 2017]. To better explain this scoring mechanism, we explicitly write out the last three transformations in the critic neural network. In (1), $h_{(x_i,\hat{y}_i)}$ is obtained by fusing the information from both raw data input $x_i$ and the predicted multi-output $\hat{y}_i$; $g(\cdot)$ is a single-value regression layer to convert the fused layer $h_{(x_i,\hat{y}_i)}$ into a single value; and the sigmoid function transforms the regressed value in the range of $[0, 1)$ as the final score. We call the $h_{(x_i,\hat{y}_i)}$ as the compatibility fusion layer of the critic network (*i.e.* the purple layer in Fig.1(a)).

The assistant function $f_A(\cdot)$ interacts with the critic network and forms a help vector $v_i = f_A([s_i, h_{(x_i,\hat{y}_i)}])$ (the green layer in Fig.1(a)) to improve the predictor. From the concatenate operation $[\cdot, \cdot]$, we know the help vector $v_i$ relies on both the critic score $s_i$ and the compatibility fusion layer $h_{(x_i,\hat{y}_i)}$ of the critic network. The reason why assistant network takes the critic score is apparent because such a score reflects the critic's final judgment. In addition, we also noticed the importance of the fusion layer $h_{(x_i,\hat{y}_i)}$ because it well depicts the compatibility of the input data $x_i$ with the current prediction $\hat{y}_i$. Compared with the single-value final score $s_i$, the layer $h_{(x_i,\hat{y}_i)}$ encodes much richer information of the criticizing process. The assistant network performs transformation $f_A(\cdot)$ to generate the help vector $v_i$ that could be directly fed into the predictor network. Consequently, the predictor takes both the raw input $x_i$ and help vector $v_i$ to make the prediction via $\hat{y}_i = f_P(x_i, v_i)$.

As shown in Fig. 1(a), the whole ACA framework is encoded in a recurrent structure. We hence unfold the recurrent ACA structure into multiple time steps in Fig.1(b). In the first time step, we initialize the help vector $v_i^{(0)}$ as zero and get the initial multi-output prediction $\hat{y}_i^{(0)}$. This initial prediction is feed-forwardly passed through both the critic and assistant modules to get a new help vector $v_i^{(1)}$. Then, we jointly feed the new help vector $v_i^{(1)}$ and raw input $x_i$ in the predictor and get the new multi-output prediction $\hat{y}_i^{(1)}$ at time point $t = 1$. Such iterative learning processes are repeated and the multi-output prediction $\hat{y}_i^{(T)}$ obtained on time point $T$ is regarded as the final prediction of the whole ACA. For the simplicity of presentation, we omit the superscript $T$ and just use $\hat{y}_i$ and $s_i$ to represent the final output of the ACA model.

## 2.1 ACA Objectives

After understanding the recurrent nature of ACA, we can now define its learning objectives. According to aforementioned discussions, there exists two types of interactions of these three modules: 1) collaborative learning between predictor (student) and assistant (TA) and 2) adversarial learning between the predictor (student) and the critic (professor). We will first define the collaborative learning loss by the two additive terms balanced

by a hyper-parameter $\lambda$:

$$\begin{aligned} L_C &= L_P(\hat{Y}, Y) + \lambda L_{CE}(S, \mathbf{1}) \\ &= L_P(\hat{Y}, Y) - \tfrac{\lambda}{N} \sum_{i=1}^{N} s_i \log s_i \end{aligned} \tag{2}$$

where the first term $L_P$ is the supervised loss (denoted by the red dot in Fig.1(a)). The predictions $\hat{Y} = [\hat{y}_1, ..., \hat{y}_N]^T$ and critic scores $S = [s_1, ..., s_N]^T$ are obtained at the final-step of the recurrent ACA (See Fig.1(b) for details). Moreover, the second term in (2) is defined from the aspect of adversarial learning. It encourages the predictor to make good predictions that can gain high scores $S$ after the criticism of the critic. We follow the same idea in GAN [Goodfellow *et al.*, 2014] to implement the adversarial objective by a cross-entropy term $L_{CE}(S, \mathbf{1})$, where $\mathbf{1}$ is a vector with all entries equal to one. From the critic's point of view, $s_i = 1$ means the predicted multi-output vector $\hat{y}_i$ is optimally compatible with the raw data $x_i$.

The above function in (2) explains the learning objective of the predictor and the assistant. We now consider the adversarial learning objective of the critic. In detail, the critic intends to assign low scores for predicted results but high scores for ground truth labels. To this end, a similar cross-entropy term is used to define the adversarial loss from the critic's perspective:

$$\begin{aligned} &L_A = L_{CE}(\{S, G\}, \{\mathbf{0}, \mathbf{1}\}) \\ &= -\tfrac{1}{N} \textstyle\sum_{i=1}^{N} (1 - s_i) \log(1 - s_i) - \tfrac{1}{N} \sum_{j=1}^{M} g_j \log g_j \end{aligned} \tag{3}$$

where $S$ are scores from predicted outputs as defined in (1) and $G = [g_1, ..., g_M]$ are compatibility scores for ground truth labels. In detail, $g_j = f_C(y_j, x_j), \forall j \in \{1...M\}$ is the critic score for the $j$th ground truth label. As reflected in (3), the loss penalizes all predicted outputs' scores $S$ to be zero but encourages ground truth scores $G$ to approximate to one. In practice, $G$ and $S$ are not necessarily to be calculated from the same set of samples $X$. We could sample a batch of $N$ samples for calculating $S$ but using another $M$ samples to calculate $G$. Such random sampling strategy allows more diverse samples' combinations in critic training. We show ACA optimization details in Algorithm 1.

In Algorithm 1, the initialization strategies for predictor $f_P$ and critic $f_C$ are task-dependent and will be clarified in the next section. Algorithm 1 also indicates that the predictor and assistant networks are updated in each iteration but the critic network is only updated every $l$ steps. This strategy facilitates the stabilization of training. We implement all three network structures with TensorFlow and adopt the ADAM optimizer [Kingma and Ba, 2014] for optimization. In inference phase, we directly pass the testing data $x_{test}$ through the recurrent structure in Fig.1 (b) to get the prediction $\hat{y}_{test}$ at the last step $T$ as the final output vector. Parameters $\theta_p$ in predictor network, $\theta_c$ in critic network and $\theta_a$ in assistant network are obtained from Algorithm 1 and can be directly used in the inference phase.

**Algorithm 1:** ACA optimization

**Input** : A training dataset $\{\mathcal{X}, \mathcal{Y}\}$; the number of recurrent ACA unfolding steps $T$; the steps $l$ for critic network updating;

**Initialization:** Initialize the predictor and critic modules.

**for** *k=1...K* **do**
- Sample a minibatch of $N$ samples $X$ and their labels $Y$ from training set $\{\mathcal{X}, \mathcal{Y}\}$;
- Initialize the help vecor $V^{(0)} = 0$ and feed-forward $X$ and $V^{(0)}$ through a $T$-step recurrent strucutre in Fig.1(b) to get the multi-output predictions $\hat{Y}$ and critic score $S = f_c(X, \hat{Y})$.
- Use $X$,$Y$,$\hat{Y}$and $S$ to calculate the loss in Eq. (2) and back-propogate the loss to update parameters $\theta_P$ in predictor network and $\theta_A$ in assistant network;
- **if** *mod(k,l)==0* **then**
  - Sample a random batch of $M$ samples $X^{(g)}$ with their ground truth labels $Y^{(g)}$;
  - Get Ground Truth evaluation score $G = f_c(X^{(g)}, Y^{(g)})$;
  - Take $G$ and $S$ in Eq.(3) for loss calculation and back-propogate the loss to update parameter $\theta_c$ in the critic network;
- **end**

**end**

**Output** : Network parameters $\theta_P$,$\theta_C$ and $\theta_A$;

## 3 The Compatibility of ACA

ACA is a general framework that is potentially compatible with many existing frameworks for multiple outputs learning. In different pplications, the structure of the assistant module is very simple and can be kept consistent across different tasks (See Section 2 for details). We hence mainly focus on the design of the predictor and the critic networks. We show details about the predictor network in Fig.1 (c) including 1) an input data encoder (red block in Fig.1 (c)) and 2) an output decoder (yellow block).

### 3.1 Multi-label Classification

Multi-label classification (MLC) requires generating a dense output vector that well describes the properties of the input data. For a $C$-class problem, the multi-label prediction result can have $2^C$ possible combinations. Existing MLC algorithms could be classified as shallow and deep approaches. In shallow configuration, label embedding projects both input data and its corresponding multi-label vector to a latent space, *e.g.* conditioned principal label space dimension reduction (CPLST) [Tai and Lin, 2012], sparse local embedding (SLEEC) [Bhatia *et al.*, 2015] and low rank empirical risk minimization for multi-label classification (LEML) [Yu *et al.*, 2014]. Deep

learning allows MLC training in a more favorable end-to-end manner. Back-propagation-based multi-label learning (BPMLL) was an early attempt to use neural networks for MLC [Zhang and Zhou, 2006]. Deep-CPLST is a deep extension of CPLST proposed in [Yeh *et al.*, 2017]. Recently, the CNN-RNN [Wang *et al.*, 2016] structure was proposed in the image attributes learning field. The canonical-correlated auto-encoder (C2AE) [Yeh *et al.*, 2017] implements a deep CCA network followed by an auto-encoder.

In the context of ACA, we can directly use these existing algorithms to fill the 'input data encoder' block in Fig.1 (c). In detail, we directly optimize different algorithms' objective functions and get their corresponding multi-label predictions as latent representations (red layer in Fig.1 (c)). Then, the predictor fuses the encoder's output with the help vector (green vector) as the input to the output decoder. In the MLC setting, the output decoder is just a multi-layer network with $C$ regressed outputs. In ACA training, we freeze all parameters in the encoder and only fine-tune parameters in the fusion and decoder blocks. Therefore, the initialization of the whole predictor can be explained in two steps. First, the encoder part can be easily initialized by running a base MLC algorithm on training data. This base algorithm can be any existing algorithm reviewed in the first paragraph of this subsection. Second, the help vector is initialized as zero and all other parameters in the fusion and decoder block are randomly initialized by a normal distribution $\mathcal{N}(0, 0.1)$.

In fact, the critic network used in ACA is exactly the same as the deep value network (DVN) discussed in [Gygli *et al.*, 2017]. By feeding both raw input data and predicted labels to a DVN, it returns the score telling how well the input data matches its label. For the consistency in presentation, we still call the DVN as a critic network in our paper. In order to pre-train the critic network, we need both compatible and incompatible pairs. A compatible sample is easily composed by pairing a data point with its ground truth label, *i.e.* $(x_i, y_i)$. Inspired by [Gygli *et al.*, 2017], we adopt two straightforward ways to generate incompatible pairs. The first approach is to match a data point with a randomly selected label *i.e.* $(x_i, y_j), j \neq i$. An alternative way is to generate an incompatible multi-label vector $\bar{y}_i$ for a data point $x_i$ by changing some entries on its ground truth label vector $y_i$. In detail, we can either remove some existing labels or add some non-existing labels to the true multi-label vector $y_i$ and get the corresponding incompatible multi-label vector $\bar{y}_i$ for $x_i$. After accumulating both compatible and incompatible pairs, it is quite easy to initialize the critic network by regressing all compatible pairs to a score 1 and incompatible pairs to score 0 with a cross entropy loss.

### 3.2 Sequence to Sequence

We also show the flexibility of ACA in dealing with sequence-to-sequence (seq2seq) tasks, which have been widely used to solve a number of natural language processing related tasks ranging from to machine translation [Kalchbrenner and Blunsom, 2013] and logic form generation [Dong and Lapata, 2016]. Seq2seq models can be easily incorporated into the encoder-decoder architecture shown in Fig.1(c). The encoder in seq2seq is mainly used to transform the input data as latent representations. Conventional encoder implementations include LSTM and bidirectional LSTM. The decoder in seq2seq is also configured by a LSTM module. It sequentially decodes each 'word' $x^t$ in a way that $x^t = LSTM(h^{t-1}, x^{t-1})$, where $h^{t-1}$ and $x^{t-1}$ are the learned hidden state and the predicted word from the last time step $t-1$. The latent vector obtained from the encoders is always used as the first hidden state $h^0$ to start the whole decoding LSTM. This strategy allows the knowledge learned from the encoder part seamlessly transmitted to the decoder side. While there are other advanced methods to implement the decoder part[Luong *et al.*, 2015], their concepts can all be well explained by this general seq2seq framework discussed above.

We implement the "input encoder" block in Fig.1 (c) by a bi-directional LSTM to enable information extracted from both directions of a sentence. Then, the latent representation is fused with the help vector forming the " initial hidden vector"to start the LSTM in the output decoder (yellow block in Fig.1(c)). Many existing LSTM decoder structures can be used here including original LSTM and its attention-based extension [Luong *et al.*, 2015]. We can easily initialize the encoding and decoding blocks in Fig.1(c) by training a normal seq2seq model with the annotated data. As before, parameters in the encoder are fixed and will not be further adjusted in the fine-tuning phase. Only parameters of the fusion and decoder parts are updated along with the end-to-end training.

In this seq2seq application, the critic takes two types of sequence data as the input for comparability evaluation. We use the English-to-Chinese machine translation task as an instance for illustration purpose. The critic takes the English sequence as one input and the translated Chinese as the other. In the critic, two LSTMs are respectively built to work English and Chinese sequences. After LSTM encoding, each sequence will be encoded as a hidden state vector as in [Kalchbrenner and Blunsom, 2013]. Then, these two hidden states are combined together forming a comparability fusion vector for critic's evaluations. To initialize the critic, we also need compatible and incompatible pairs as before. In detail, for each input sequence (English sentence), we build a compatible pair by matching it with its annotated ground truth translation (the authentic Chinese translation). On the other hand, we can simply generate some incompatible pairs by replacing removing or exchanging some random words on the annotated output sentence to get a wrong output (a wrongly translated Chinese sentence). This wrong output will be paired with the input sequence (English sentence) to compose an incompatible pair. The critic is trained with these compatible and incompatible pairs by a cross-entropy loss.

| | Bibtex | Bookmarks | Delicious |
|---|---|---|---|
| CPLIST | 37.5 ± 0.5 | 30.5 ± 0.5 | 33.6 ± 0.4 |
| ACA-CPLIST | 40.8 ± 0.4 | 33.4 ± 0.6 | 36.4 ±0.4 |
| SLEECE | 39.1± 0.6 | 31.2 ± 0.7 | 35.9 ± 0.6 |
| ACA-SLEECE | 42.2± 0.5 | 33.9± 0.6 | 37.5± 0.6 |
| DEEP-CPLIST | 38.8± 0.4 | 31.7± 0.4 | 37.2± 0.5 |
| ACA-D-C | 40.5± 0.5 | 33.8± 0.6 | 38.6± 0.4 |
| BPMIL | 40.9± 0.7 | 32.1± 0.5 | 38.3± 0.6 |
| ACA-BPMIL | 43.7± 0.5 | 34.9± 0.5 | **40.2± 0.6** |
| CA2E | 41.6± 0.7 | 33.5± 0.7 | 37.8± 0.6 |
| ACA-CA2E | **44.3± 0.6** | **35.6± 0.7** | 39.3± 0.6 |

Table 1: F scores evaluated on benchmark datasets (%)

# 4 Experiments

## 4.1 Documents Modeling

We evaluate the performances of ACA on multiple-label classification (MLC) for documents modeling on 'bibtex' and 'bookmark' datasets[Loza Mencía and Fürnkranz, 2008]. The bibtex dataset contains $7,395$ samples from 159 classes; and bookmark dataset contains $87,856$ samples within 208 classes. We also include the 'delicious' dataset [Tsoumakas *et al.*, 2008] in our experiment that contains $16,105$ samples in 983 classes. The label vector for each sample exhibits multi-classes association. For instance, there are $2^{159}$ possible label combinations for a sample in the bibtex dataset. We will investigate how performances of existing methods could be improved based on the ACA framework. To this end, we select both prevalent shallow and deep methods as building blocks incorporated into ACA.

We randomly sample 20% of the whole data for testing and the other 80% data are for training and validation. The random splitting, training and testing processes are repeated for 10 times. Different methods' performances (original and ACA improved results) on these three multi-label datasets have been provided in Table **??**. The details about these MLC methods have been discussed in section **??**. Following existing works, we use the Macro F1 score as the accuracy indicator. In each row-wise block of Table **??**, the above line summarizes the results from the original MLC method and the second row reports results of the same algorithm enhanced by ACA. In ACA, the dimensions for help vector $v_i$ and comparability fusion layer $h_{(x_i,y_i)}$ are both fixed as 64. The recurrent steps $T$ is fixed as 5.

From experimental comparisons, we can get to the following conclusions. First, with either shallow or deep MLC method as the building block, ACA could significantly improve the original method's accuracy. Second, deep MLC methods greatly outperform the shallow ones. Finally, ACA is also very effective to improve performances on the challenging dataset with thousands of labels (as observed from 'delicious' dataset).

## 4.2 Image Attributes Prediction

In this image test, we only consider deep-learning-based MLC methods on the NUS-wide [Chua *et al.*, July 8 10 2009], ESP-game [Guillaumin *et al.*, 2009] and CUB-bird datasets[Wah *et al.*, 2011] datasets. Following [Yeh *et al.*, 2017; Deng *et al.*, 2019], we randomly select $r$ data points in each dataset for training/validation and the rest are for testing purpose. The average Micro/Macro-F1 score reported in Table 2 that were calculated from 10 replicates. We also consider CNN-RNN [Wang *et al.*, 2016] and WARP [Gong *et al.*, 2013] for comparisons which are two prevalent attributes prediction methods for images.

We further consider variations of ACA implementations. The first variation is the adversarial critic model (AC) [Deng *et al.*, 2019] where the assistant module is removed from the ACA. Then, we consider ACA with a single score feedback (ACS) that we directly feed the final critic score as the assistant feedback to the predictor. The results of these variations are listed in the last two columns of Table 2. We chose the BPMLL model as the basic predictor here due to its simplicity and effectiveness. From the results, we have observed that AC gains improvements over the original BPMILL method. The performance by ACS is not that different than the AC method without feedback. However,the ACA outperforms all competing adversarial methods on these three datasets.

## 4.3 Logic Form Generation from Utterance

Converting human language into executable logic forms are a fundamental problem in natural language understanding (NLU). For instance, the sentence "*nonstop flight ci1 to ci0*" can be converted to a logic form "*(lambda $0 e (and(flight $0)(nonstop $0)(from $0 ci1)(to $0 ci0)))*". This logic form generation task is well tackled by seq2seq leaning. Here, we consider two widely used datasets including ATIS (5410 queries to a flight booking system) [Hemphill *et al.*, 1990] and GEO (880 queries about U.S. geography) [Dong and Lapata, 2016] to conduct our experiments. In these datasets, we follow the augmentation identification approach [Dong and Lapata, 2016] to replace entities and numbers in input with their underlying type names and unique IDs. For instance, in the aforementioned instance, we replace city names with symbols 'c1' and 'c2'. To conduct seq2seq learning, words of input sequence and symbols of the output sequence are both converted as a series of one-hot vectors. The 'hot' entry indicates the certain class of a word/symbol. We follow models introduced in Section 3.2 to implement the seq2seq model of ACA. Both the LSTM and the attention model are considered as the decoder. For the attention model, we use the same structure introduced in [Dong and Lapata, 2016]. In this work, all LSTM are implemented with 128 hidden states.

In each dataset, we randomly sample 80% sentences for training, 10% for parameter validation and the rest 10% for testing. This random processes are repeated for

| Data | | CNN-RNN | | Deep-CPLST | | WARP | | BPMLL | | CA2E | | Adversarial | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | Ori. | ACA | Ori. | ACA | Ori. | ACA | Ori. | ACA | Ori. | ACA | AC | ACS |
| NUS | MicF1 | 33.7 | 36.2 | 35.4 | 37.5 | 32.7 | 37.8 | 37.9 | 48.2 | 47.9 | 49.4 | 41.7 | 42.3 |
| | MacF1 | 54.8 | 57.9 | 56.9 | 60.3 | 53.5 | 56.2 | 61.8 | 67.0 | 67.3 | 68.2 | 64.5 | 63.8 |
| ESPE | MicF1 | 11.2 | 13.1 | 5.4 | 9.8 | 6.4 | 10.2 | 13.9 | 14.8 | 13.6 | 14.9 | 14.4 | 13.9 |
| | MacF1 | 17.8 | 20.1 | 9.3 | 17.1 | 6.5 | 16.0 | 19.2 | 25.7 | 21.3 | 24.6 | 23.4 | 23.2 |
| CUB | MicF1 | 6.3 | 7.3 | 6.2 | 9.5 | 6.1 | 9.2 | 7.8 | 10.5 | 8.8 | 10.5 | 9.5 | 9.4 |
| | MacF1 | 15.3 | 16.4 | 13.9 | 16.8 | 14.3 | 16.5 | 17.8 | 21.3 | 15.8 | 19.1 | 18.6 | 18.9 |

Table 2: The performance of ACA on image datasets evaluated by micro-F1 and macro-F1 measures

| Categories | | ATIS | GEO |
|---|---|---|---|
| LSTM | Original | 81.9±0.9 | 80.4±0.7 |
| | ACA | 83.1±0.7 | 83.6±0.6 |
| Attention | Original | 82.0±1.1 | 81.7±0.8 |
| | ACA | **85.2±0.9** | **84.1±0.8** |
| Adversarial | AC | 82.9±0.8 | 83.1±0.7 |
| | ACS | 83.6±0.9 | 82.4±0.7 |
| Critic | SeqGAN | 82.7 ± 0.9 | 83.4 ± 0.8 |
| | Self-Critic | 83.2 ± 0.7 | 81.9 ± 0.8 |
| | Actor-Critic | 83.8 ± 0.6 | 82.9 ± 0.7 |

Table 3: Accuracy of the logic form generation task %

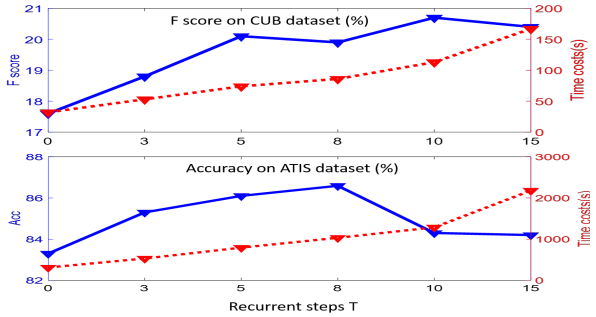



Figure 2: ACA learning performances and computational complexity with different recurrent steps $T$.

10 times with results summarized in Table 3. In the table, ACA gains apparent improvements on both LSTM-based and attention-based seq2seq models. The results of ACA excluding the assistant module (AC) and ACA with a single score feedback (ACS) are also provided for references. We also consider using the default data splitting in [Dong and Lapata, 2016] to verify the performance of ACA. In such a case, attention-based ACA achieves 86.7% and 86.4% on ATIS and GEO datasets which are 2.1% and 1.8% higher than the reported attention model results in [Dong and Lapata, 2016].

We further compare ACA with alternative critic networks for sequence generation. Here, the competitors include SeqGAN [Yu *et al.*, 2017], self-critic [Rennie *et al.*, 2016] and actor-critic [Bahdanau *et al.*, 2017]. SeqGAN borrows the concept of reinforcement learning to sample new sentences in a Monte Carlo manner. Self-critic method uses the prevalent REINFORCE algorithm to perform policy gradient learning. The actor-critic framework incorporates both an actor and a critic networks for sequence prediction in a supervised manner. We report results of these algorithms in Table 3 (see the Critic category). It is shown that all these critic-based methods get much better results than the original approaches in both LSTM and Attention categories. However, none of these complicated reinforcement-learning-based critic could beat the performance of ACA. It hence verifies the effectiveness of the assistant module in ACA, which is the unique ingredient of our approach.

### 4.4 ACA with Different Recurrent Steps

We report the learning performance and complexity of ACA with different steps $T$ for both the MLC (CUB dataset) and the seq2seq (ATIS dataset) tasks. In CUB (*resp.* ATIS) dataset, ACA is equipped with the BPMIL (*resp.* attention LSTM) as the base model. In Fig.2, blue curves (with scalers on the left y-axis) summarize the algorithm performance; and red curves (with scalers on the right y-axis) report the computational costs. All reported time is calculated by running our algorithm with TensorFlow on 8 GPUs. $T = 0$ means running ACA without assistant module involved. We have observed that a large $T$ number will significantly increase the computational costs but not the performances.Therefore, we suggest choose $T = 5$ as the default recurrent steps in all previous experiments.

## 5 Discussions

We proposed the ACA framework to improve existing learning algorithms with multiple outputs. It is a general framework that could be robustly applied to many tasks across different domain. We have also tested ACA on other single-output classification and regression tasks. Unfortunately, no significant improvements were observed on those tasks. This limitation may be due to the simplicity of the output structure, where less information could be criticized by the critic. In the future, we will consider improvements on the current ACA framework to make it also adapted to single-output tasks.

## References


[Bahdanau *et al.*, 2017] Dzmitry Bahdanau, Philemon Brakel, Kelvin Xu, Anirudh Goyal, Ryan Lowe, Joelle Pineau, Aaron Courville, and Yoshua Bengio. An actor-critic algorithm for sequence prediction. *ICLR*, 2017.

[Bhatia *et al.*, 2015] Kush Bhatia, Himanshu Jain, Purushottam Kar, Manik Varma, and Prateek Jain. Sparse local embeddings for extreme multi-label classification. In *NIPS*, pages 730–738, 2015.

[Chua *et al.*, July 8 10 2009] Tat-Seng Chua, Jinhui Tang, Richang Hong, Haojie Li, Zhiping Luo, and Yan-Tao. Zheng. Nus-wide: A real-world web image database from national university of singapore. In *CIVR*, Santorini, Greece., July 8-10, 2009.

[Deng *et al.*, 2019] Y. Deng, K. Chen, Y. Shen, and H. Jin. Adversarial multi-label prediction for spoken and visual signal tagging. In *ICASSP 2019 - 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 3252–3256, May 2019.

[Dong and Lapata, 2016] Li Dong and Mirella Lapata. Language to logical form with neural attention. *ACL*, 2016.

[Gong *et al.*, 2013] Yunchao Gong, Yangqing Jia, Thomas Leung, Alexander Toshev, and Sergey Ioffe. Deep convolutional ranking for multilabel image annotation. *arXiv:1312.4894*, 2013.

[Goodfellow *et al.*, 2014] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *NIPS*, pages 2672–2680, 2014.

[Guillaumin *et al.*, 2009] M. Guillaumin, T. Mensink, J. Verbeek, and C. Schmid. Tagprop: Discriminative metric learning in nearest neighbor models for image auto-annotation. In *ICCV*, pages 309–316, Sept 2009.

[Gygli *et al.*, 2017] Michael Gygli, Mohammad Norouzi, and Anelia Angelova. Deep value networks learn to evaluate and iteratively refine structured outputs. *ICML*, 2017.

[Hemphill *et al.*, 1990] Charles T Hemphill, John J Godfrey, and George R Doddington. The atis spoken language systems pilot corpus. In *Speech and Natural Language: Proceedings of a Workshop Held at Hidden Valley, Pennsylvania, June 24-27, 1990*, 1990.

[Kalchbrenner and Blunsom, 2013] Nal Kalchbrenner and Phil Blunsom. Recurrent continuous translation models. In *EMNLP*, volume 3, page 413, 2013.

[Kingma and Ba, 2014] Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv:1412.6980*, 2014.

[Lin *et al.*, 2014] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *ECCV*, pages 740–755. Springer, 2014.

[Loza Mencía and Fürnkranz, 2008] Eneldo Loza Mencía and Johannes Fürnkranz. Efficient pairwise multilabel classification for large-scale problems in the legal domain. *Machine Learning and Knowledge Discovery in Databases*, pages 50–65, 2008.

[Luong *et al.*, 2015] Minh-Thang Luong, Hieu Pham, and Christopher D Manning. Effective approaches to attention-based neural machine translation. *EMNLP*, 2015.

[Rennie *et al.*, 2016] Steven J Rennie, Etienne Marcheret, Youssef Mroueh, Jarret Ross, and Vaibhava Goel. Self-critical sequence training for image captioning. *arXiv:1612.00563*, 2016.

[Tai and Lin, 2012] Farbound Tai and Hsuan-Tien Lin. Multilabel classification with principal label space transformation. *Neural Computation*, 24(9):2508–2542, 2012.

[Tsoumakas *et al.*, 2008] Grigorios Tsoumakas, Ioannis Katakis, and Ioannis Vlahavas. Effective and efficient multilabel classification in domains with large number of labels. In *ECML/PKDD*, pages 30–44, 2008.

[Vinyals *et al.*, 2015] Oriol Vinyals, Alexander Toshev, Samy Bengio, and Dumitru Erhan. Show and tell: A neural image caption generator. In *CVPR*, pages 3156–3164, 2015.

[Wah *et al.*, 2011] Catherine Wah, Steve Branson, Peter Welinder, Pietro Perona, and Serge Belongie. The caltech-ucsd birds-200-2011 dataset. 2011.

[Wang *et al.*, 2016] Jiang Wang, Yi Yang, Junhua Mao, Zhiheng Huang, Chang Huang, and Wei Xu. Cnn-rnn: A unified framework for multi-label image classification. In *CVPR*, pages 2285–2294, 2016.

[Yeh *et al.*, 2017] Chih-Kuan Yeh, Wei-Chieh Wu, Wei-Jen Ko, and Yu-Chiang Frank Wang. Learning deep latent spaces for multi-label classification. 2017.

[Yu *et al.*, 2014] Hsiang-Fu Yu, Prateek Jain, Purushottam Kar, and Inderjit Dhillon. Large-scale multi-label learning with missing labels. In *ICML*, pages 593–601, 2014.

[Yu *et al.*, 2017] Lantao Yu, Weinan Zhang, Jun Wang, and Yong Yu. Seqgan: Sequence generative adversarial nets with policy gradient. In *AAAI*, pages 2852–2858, 2017.

[Zhang and Zhou, 2006] Min-Ling Zhang and Zhi-Hua Zhou. Multilabel neural networks with applications to functional genomics and text categorization. *TKDE*, 18(10):1338–1351, 2006.