DEBUG: A Dense Bottom-Up Grounding Approach for Natural Language Video Localization

Chujie Lu[1*] **Long Chen**[1*] **Chilie Tan**[2] **Xiaolin Li**[2,3] **Jun Xiao**[1†]
[1]DCD Lab, College of Computer Science, Zhejiang University, China
[2]Tongdun Technology, China [3]University of Florida, USA
chujielu@outlook.com, {longc, junx}@zju.edu.cn
chilie.tan@tongdun.net, andyli@ece.ufl.edu

Abstract

In this paper, we focus on natural language video localization: localizing (*i.e.*, grounding) a natural language description in a long and untrimmed video sequence. All currently published models for addressing this problem can be categorized into two types: (i) top-down approach: it does classification and regression for a set of pre-cut video segment candidates; (ii) bottom-up approach: it directly predicts probabilities for each video frame as the temporal boundaries (*i.e.*, start and end time point). However, both two approaches suffer several limitations: the former is computation-intensive for densely placed candidates, while the latter has trailed the performance of the top-down counterpart thus far. To this end, we propose a novel *dense bottom-up* framework: DEnse Bottom-Up Grounding (DEBUG). DEBUG regards all frames falling in the ground truth segment as foreground, and each foreground frame regresses the unique distances from its location to bi-directional ground truth boundaries. Extensive experiments on three challenging benchmarks (TACoS, Charades-STA, and ActivityNet Captions) show that DEBUG is able to match the speed of bottom-up models while surpassing the performance of the state-of-the-art top-down models.

1 Introduction

Vision-and-language understanding, *e.g.*, what the vision and text are, and how they relate with each other, is one of the core tasks in both computer vision and natural language processing. To test the machine comprehension of complex video scene and natural language simultaneously, a challenging task was proposed (Gao et al., 2017; Hendricks et al., 2017, 2018), called **Natural Language Video Localization** (NLVL). As shown in Figure 1, given a natural language description query and an untrimmed video sequence, NLVL needs to localize a segment in the video (*i.e.*, identify the start and end time point of the segment) which semantically corresponds to the reference sentence. Moreover, NLVL is an indispensable technique for many important applications, *e.g.*, text-oriented video highlight detection or retrieval.

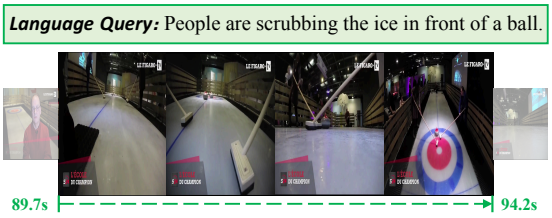

Figure 1: Natural language video localization is to localize a segment with the start point (89.7s) and end point (94.2s) in the video given a language description.

Currently, the overwhelming majority of NLVL models are ***top-down*** approaches: they first cut a video into a set of segment candidates, then they do classification and regression for each candidate. Specifically, they can be further grouped into two sub-types: 1) *sliding-window-based* (Gao et al., 2017; Hendricks et al., 2017; Liu et al., 2018b,a; Ge et al., 2019; Chen and Jiang, 2019; Xu et al., 2019; Zhang et al., 2019b): the video is explicitly segmented by multiple predefined temporal sliding-window scales. After extracting features for the query and all candidates, NLVL degrades into a multimodal matching problem. 2) *anchor-based* (Chen et al., 2018; Zhang et al., 2019a): it assigns each frame[1] with multi-scale temporal anchors, which follows the same spirit of anchor box in object detection (Ren et al., 2015).

Although top-down approaches have dominated NLVL for years, it is worth noting that they suffer

[*] Chujie Lu and Long Chen are co-first authors with equal contributions.

[†] Jun Xiao is the corresponding author.

[1]The frame is a general description for a frame in a video sequence or an element in a video frame feature sequence.

Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, pages 5144–5153, Hong Kong, China, November 3–7, 2019.

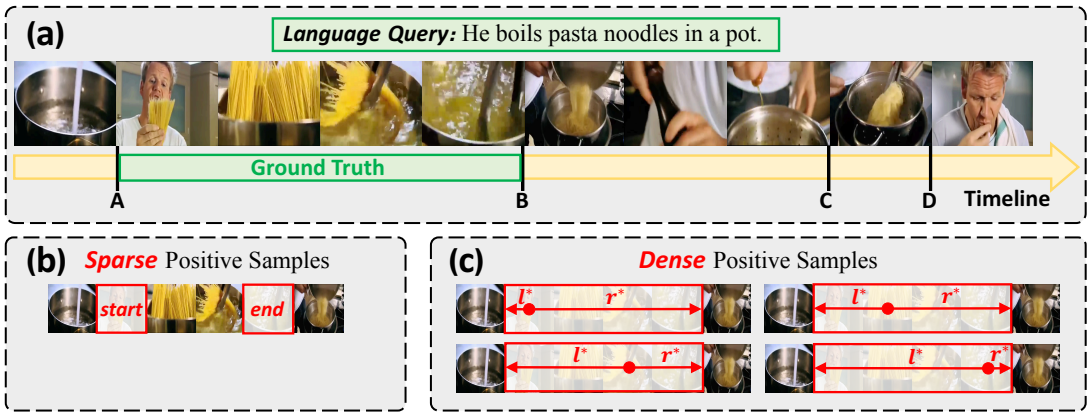

Figure 2: (a): A NLVL sample with ground truth segment (A→B). (b): Sparse positive samples in existing sparse bottom-up models (*i.e.*, two frames). (c): Dense positive samples in DEBUG (*i.e.*, all frames in range (A→B)).

several notorious limitations: 1) The performance is sensitive to the heuristic rules (*e.g.*, the temporal scales or the number of candidates). 2) In order to achieve a high recall, they are required to densely place candidates, which significantly increase the amount of computation and localization time.

To eliminate these inherent drawbacks in top-down framework, some recent NLVL works (Chen et al., 2019a; Yuan et al., 2019) start to borrow idea from reading comprehension (Xiong et al., 2017, 2018; Yu et al., 2018), which directly predicts the start and end boundaries. Although this ***sparse bottom-up*** approach is highly computation-efficient, the localization accuracy, especially for long videos (*e.g.*, TACoS) is behind its top-down counterpart. We argue that the main reasons come from three-fold: 1) Two boundary predictions are independent, *i.e.*, the model ignores the content consistency between two predictions. As an example shown in Figure 2 (a), two frames in B and D have a similar visual appearance. Thus, the model is prone to predict the result as (A→D), without considering the distinct content change in range (B→C). 2) The positive and negative training samples are extremely imbalanced. Since the number of video frames is large (*e.g.*, each video in TACoS has average 9,000 frames), but the positive training samples are *sparse*, *i.e.*, only two frames (Figure 2 (b)). 3) Detecting temporal action boundary from frames, *i.e.*, predicting a frame is query-related and at temporal boundary simultaneously by a single network, is still a challenging task, even without query constraint (Shou et al., 2018).

In this paper, we propose a ***dense bottom-up*** framework for NLVL: DEnse Bottom-Up Grounding (DEBUG), to mitigate the problems in existing NLVL frameworks. Specifically, we regard all frames falling in the ground truth segment as positive samples (*i.e.*, foreground). For each positive frame, DEBUG has a classification subnet to predict its relatedness with the query, and a boundary regression subnet to regress the unique distances from its location to bi-directional ground truth boundaries. This design helps to disentangle the temporal boundary detection from query-related prediction, relieving the burden of the single classification network in existing sparse bottom-up models. Meanwhile, we can utilize as many positive samples as possible to alleviate the imbalance problem between positive and negative samples (Figure 2 (c)). Since each pair of boundary predictions are based on the same frame feature, *i.e.*, two predictions act as a whole, which helps to avoid falling into the local optimum caused by independent predictions. In addition, we propose a temporal pooling to relieve unstable performance caused by single prediction. Moreover, DEBUG is agnostic to the upstream multimodal interaction network, *i.e.*, it can be seamlessly incorporated into any stronger backbone to boost performance.

We demonstrate the effectiveness of DEBUG on three challenging benchmarks: TACoS (Regneri et al., 2013), Charades-STA (Gao et al., 2017), and ActivityNet Captions (Krishna et al., 2017). Without bells and whistles, DEBUG surpasses the performance of the state-of-the-art models over various benchmarks and metrics at the highest speed.

2 Related Work

2.1 Natural Language Video Localization

NLVL is a very difficult task, which needs to understand both complex video scene and natu-

ral language simultaneously. Because most of NLVL models are under the top-down framework, they focus on designing more effective multi-modal interaction networks, *e.g.*, query-based attention on video frames (Liu et al., 2018a), visual-based attention on language words (Liu et al., 2018b), or co-attention between each frame-and-word pairs (Chen et al., 2018, 2019a; Yuan et al., 2019). It is worth noting that the improvement in multimodal interaction network is orthogonal to the DEBUG, *i.e.*, DEBUG can be seamlessly incorporated into any stronger interaction network.

To the best of our knowledge, there are only two exceptions among all NLVL models: RWM (He et al., 2019) and SM-RL (Wang et al., 2019), which are not under either top-down or bottom-up frameworks. They both formulate NLVL as a sequential decision making problem, solved by reinforcement learning, *e.g.*, actor critic (Chen et al., 2019b). The action space for each step is a set of handcraft-designed temporal box transformations.

2.2 Top-Down vs. Bottom-Up

Top-down and bottom-up approaches, which are widely co-exist in many CV and NLP tasks, are two different philosophical viewpoints for solving problems. The most related top-down and bottom-up concepts as the one in NLVL frameworks are:

Object Detection. Most of the object detectors after Faster-RCNN (Ren et al., 2015) are top-down models, *i.e.*, it predicts classification scores and regression offsets for multiple predefined anchors in each position. These models suffer the same drawbacks as above mentioned in the top-down approach for NLVL. However, with the advent of the first performance comparable bottom-up object detector: CornerNet (Law and Deng, 2018), the bottom-up approaches begin to gain unprecedented attention (Zhou et al., 2019b,a; Duan et al., 2019; Tian et al., 2019), which inspires us to explore a decent bottom-up framework for NLVL.

Attention Mechanism. Top-down attention has dominated many vision-and-language tasks, *e.g.*, visual captioning (Xu et al., 2015; Chen et al., 2017), visual QA (Xu and Saenko, 2016; Ye et al., 2017). Recently, a model combining both top-down and bottom-up attention reaches the winner of multiple challenges (Anderson et al., 2018). Thus, how to combine the top-down and bottom-up attentions effectively is still an unexplored problem in the vision-and-language tasks.

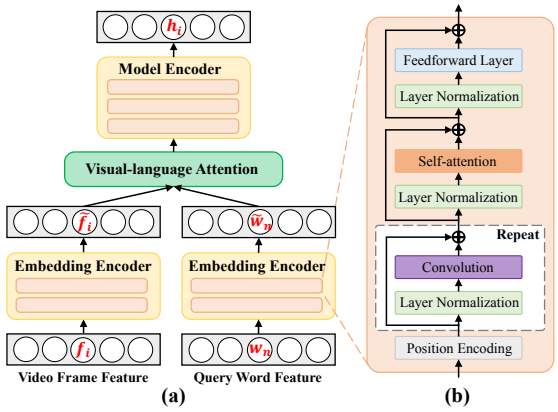

Figure 3: (a): The overview of QANet backbone. (b): The encoder block throughout the QANet.

3 Approach

The NLVL task considered in this paper, is defined as follows. Given a long and untrimmed video sequence V, and a natural language query Q which describes a segment in V from start time point t_s to end time point t_e, NLVL needs to predict these two time points (t_s, t_e) given V and Q.

In this section, we first introduce the multi-modal interaction backbone of DEBUG, which is built upon the recently proposed QANet (Yu et al., 2018) for reading comprehension (Section 3.1). Then, we demonstrate the details about the proposed dense bottom-up grounding (Section 3.2). Finally, we describe the training and test stage of the whole DEBUG (Section 3.3).

3.1 Backbone: QANet

We adopt the QANet to model the interaction between two different modalities (*i.e.*, video and language), which serves as the backbone of DEBUG. The details of the QANet are shown in Figure 3 (a), which consists of three main components:

Embedding Encoder Layer. The input for this layer in two different branches are extracted video frame features $\boldsymbol{F} = \{\boldsymbol{f}_i\}_{i=1}^{T}$ and query word features $\boldsymbol{W} = \{\boldsymbol{w}_n\}_{n=1}^{N}$, respectively (see details in Section 4.2). T and N are the numbers of frames and words. The embedding encoder layer is a stack of encoder blocks as shown in Figure 3 (b), which contains multiple components, including convolutional layer, layer-normalization layer, self-attention layer, and feedforward layer. The output of this layer is new frame features $\tilde{\boldsymbol{F}} = \{\tilde{\boldsymbol{f}}_i\}_{i=1}^{T}$ or word features $\tilde{\boldsymbol{W}} = \{\tilde{\boldsymbol{w}}_n\}_{n=1}^{N}$, which encode the context in its respective modality.

Visual-language Attention Layer. It calculates

two different attention weights between the two modal features. Specifically, it first computes a similarity matrix $\boldsymbol{S} \in \mathbb{R}^{T\times N}$, where $\boldsymbol{S}_{ij}$ denotes the similarity between frame feature $\tilde{\boldsymbol{f}}_i$ and word feature $\tilde{\boldsymbol{w}}_j$. Then the two attention weights are:

$$\boldsymbol{A} = \bar{\boldsymbol{S}} \cdot \tilde{\boldsymbol{W}}, \quad \boldsymbol{B} = \bar{\boldsymbol{S}} \cdot \bar{\bar{\boldsymbol{S}}}^T \cdot \tilde{\boldsymbol{F}}, \tag{1}$$

where $\bar{\boldsymbol{S}}$ and $\bar{\bar{\boldsymbol{S}}}$ are the row-wise and column-wise normalized matrix of S, respectively.

Model Encoder Layer. Given the two attention weights $\boldsymbol{A}$ and $\boldsymbol{B}$, the model encoder layer begins to encode the interaction between the two modal features. This layer is also a stack of encoder blocks (Figure 3 (b)), and these encoder blocks share parameters. The input in i-th position is $[\boldsymbol{f}_i, \boldsymbol{a}_i, \boldsymbol{f}_i \odot \boldsymbol{a}_i, \boldsymbol{f}_i \odot \boldsymbol{b}_i]$, where $\boldsymbol{a}_i$ and $\boldsymbol{b}_i$ are i-th row of $\boldsymbol{A}$ and $\boldsymbol{B}$, $\odot$ is the element-wise multiplication, and $[\cdot]$ is a vector concatenate operation. The output is $\boldsymbol{H} = \{\boldsymbol{h}_i\}_{i=1}^T, \boldsymbol{H} \in \mathbb{R}^{T\times D}$, where $\boldsymbol{h}_i \in \mathbb{R}^D$ is a frame feature encoded with multimodal context. Then $\boldsymbol{H}$ is fed into the following head network (*i.e.*, dense bottom-up grounding) for boundaries prediction. We refer readers to QANet (Yu et al., 2018) paper for more details.

3.2 Dense Bottom-Up Grounding

Since the nature of the bottom-up approach, DEBUG regards each frame as a training sample. Different from the existing sparse bottom-up models which only use the exact start and end boundary frames as foreground, DEBUG utilizes all frames falling in the ground truth segment as positive samples. For each sample, there are three branch subnets, which aim to predict its classification score, boundary distances, and confidence score respectively. Specifically, the whole architecture of DEBUG is shown in Figure 4, and the details about the three branch subnets are:

Classification Subnet. The classification subnet predicts the relatedness between each video frame and the language query, *i.e.*, whether the frame is a foreground frame. Taking the multimodal feature $\boldsymbol{H} \in \mathbb{R}^{T\times D}$ from the backbone, this subnet applies four 1×3 conv layers, each with D filters and each followed by ReLU activations, followed by a 1×3 conv layer with 1 filter. Finally, sigmoid activations are attached to output the foreground prediction score per location. For a positive sample (*i.e.*, foreground), its ground truth classification label is $c_i^* = 1$, otherwise $c_i^* = 0$.

Boundary Regression Subnet. The boundary regression subnet predicts the unique distances from

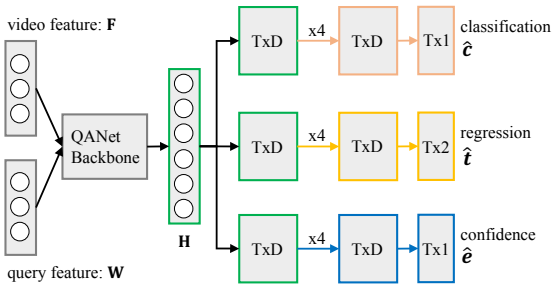

Figure 4: The whole architecture of the DEBUG. It consists of a QANet backbone and a head network with three branch subnets. $T \times *$ denotes the shape of corresponding matrix.

the location of each frame to the bi-directional ground truth boundaries. The design of the boundary regression subnet is identical to the classification subnet except it terminates in 2 outputs for both left and right distances. We only assign boundary regression targets for positive frames. Specifically, for a positive frame at i-th position, if the ground truth segment range is (t_s, t_e) (*i.e.*, $t_s \le i \le t_e$), the regression target is $\boldsymbol{t}_i^* = (l_i^*, r_i^*)$:

$$l_i^* = i - t_s, \quad r_i^* = t_e - i, \tag{2}$$

where l_i^* and r_i^* represents the distance from i-th frame to the left and right boundaries, respectively.

Confidence Subnet. The design of the confidence subnet is identical to the classification subnet, but it predicts the confidence of the boundary regression results for each frame. The motivation of this subnet design comes from that the prediction confidences from different frames should be different, *e.g.*, it is more difficult for a frame near the start point to detect the end point than a frame near the end point. Therefore, we set the ground truth confidence of each frame based on its "centerness" in the segment. Given the regression target l_i^* and r_i^*, the ground truth confidence score is defined as:

$$e_i^* = \frac{\min(l_i^*, r_i^*)}{\max(l_i^*, r_i^*)}. \tag{3}$$

The confidence score ranges from 1 to 0 with the frame position from segment center to boundary.

3.3 Training and Inference

Loss. Given all frames predictions $\{(\hat{c}_i, \hat{\boldsymbol{t}}_i, \hat{e}_i)\}$ and the corresponding ground truth $\{(c_i^*, \boldsymbol{t}_i^*, e_i^*)\}$, the total training loss function for DEBUG is:

$$L = \frac{1}{N}\sum_i L_{cls}(\hat{c}_i, c_i^*) + \frac{\alpha}{N_p}\mathbb{1}_{\{c_i^*=1\}}L_{reg}(\hat{\boldsymbol{t}}_i, \boldsymbol{t}_i^*) + \frac{\beta}{N_p}\mathbb{1}_{\{c_i^*=1\}}L_{conf}(\hat{e}_i, e_i^*) \tag{4}$$

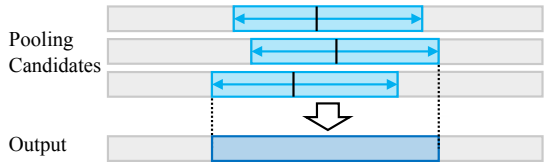

Figure 5: Illustration of temporal pooling.

where L_{cls} and L_{conf} both are binary cross entropy (BCE) loss for classification subnet and confidence subnet. L_{reg} is the IOU loss for boundary regression subnet (*i.e.*, $-\ln \frac{\min(r_i^*,\hat{r}_i)-\max(l_i^*,\hat{l}_i)}{\max(r_i^*,\hat{r}_i)-\min(l_i^*,\hat{l}_i)}$). N and N_p denotes the number of total samples and positive samples, respectively. α and β are loss weights to balance different losses, we set both α and β to 1 in all experiments. $\mathbb{1}_{\{c_i^*=1\}}$ is an indicator function, being 1 if $c_i^* = 1$ and 0 otherwise.

Inference. Given a video and a language query, we forward them through the network and obtain $\hat{c}_i$, $\hat{\boldsymbol{t}}_i$, $\hat{e}_i$ for each frame from three subnets. Then, we rank all segment predictions by the score $\hat{s}$: $\hat{s}_i = \hat{c}_i \times \hat{e}_i$. A straightforward solution is selecting the segment with the highest score as the final prediction. However, segment prediction from a single frame is usually with high variance. To relieve this situation, we propose a simple yet effective strategy: **Temporal Pooling**, to fuse multiple frame predictions. As shown in Figure 5, temporal pooling directly uses the leftmost and rightmost boundaries among all pooling candidates as its output. As for the pooling candidates, they need to meet with two conditions simultaneously: 1) the predicted segment is overlapped with the one with the highest score; 2) the score is large than the highest score multiple a threshold δ[2].

4 Experiments

4.1 Datasets and Metrics

Datasets. We evaluated the DEBUG on three challenging NLVL benchmarks: **TACoS** (Regneri et al., 2013). It consists of 127 videos and 17,344 text-to-clip pairs. In our experiments, we used the standard split same as (Gao et al., 2017), *i.e.*, 50% for training, 25% for validation and 25% for test. The average length of each video is 5 minutes. **Charades-STA** (Gao et al., 2017). It consists of 12,408 text-to-clip pairs for training, and 3,720 pairs for test. The average length of each video is 30 seconds. **ActivityNet Captions** (Krishna et al., 2017). It is not only the largest NLVL dataset (19,209 videos) but also with much more diverse context than the others. We followed (Yuan et al., 2019) and utilized the public train set (*i.e.*, 37,421 text-to-clip pairs) for training, and the validation set (*i.e.*, 17,505 text-to-clip pairs) for test. The average length of each video is 2 minutes.

Evaluation Metrics. Following the conventions in previous works, we evaluated NLVL on two prevailing evaluation metrics: 1) **R@N, IoU@θ**: The percentage of testing samples which have at least one of the top-N results with IoU larger than θ. Since the nature of the bottom-up framework, we only use N=1 in all our experiments; 2) **mIoU**: The average IoU over all testing samples.

4.2 Implementation Details

Given an untrimmed video V, we first downsampled frames and utilized the C3D (D.Tran et al., 2015) feature pretrained on Sports-1M (A.Karpathy et al., 2014) as the initial frame features. Then, we reduced the dimension of these features to 500 using PCA, which are the video frame features $\boldsymbol{F}$ (Section 3.1). For query Q, it was truncated or padded to a maximum length of 15 words. Each word was initialized with the 300-d Glove vector (J.Pennington et al., 2014), and all word embeddings were fixed. Then we learned a transformation matrix to map these embeddings to 500-d, which are the sentence word features $\boldsymbol{W}$ (Section 3.1). The dimension of all intermediate layers in the backbone and three subnets was set to 128. We trained the whole network for 100 epochs from scratch, and the loss was optimized by Adam algorithm (D.P.Kingma and J.Ba, 2015). The learning rate started from 0.0001 and was divided by 10 when the loss plateaus. The batch size was set to 16, and the dropout rate was 0.5.

4.3 Comparisons with State-of-the-Arts

Settings. We compared the DEBUG with all recently published state-of-the-art NLVL models. From the viewpoint of top-down and bottom-up framework, we group them into: 1) Sliding-window-based models: **VSA-RNN**, **VSA-STV**, **CTRL** (Gao et al., 2017), **ROLE** (Liu et al., 2018b), **ACRN** (Liu et al., 2018a), **MCF** (Wu and Han, 2018), **ACL** (Ge et al., 2019), **SAP** (Chen and Jiang, 2019), **QSPN** (Xu et al., 2019). 2) Anchor-based models: **TGN** (Chen et al., 2018). 3) Sparse bottom-up models: **L-Net** (Chen et al., 2019a), **ABLR-af**, **ABLR-aw** (Yuan et al., 2019)

[2]In experiments, we tested $\delta \in \{0.1, 0.2, ..., 0.9\}$ and selected the one with the highest performance for each dataset.

	Method	IoU@0.1	IoU@0.3	mIoU
TACoS	VSA-RNN	8.84	6.91	-
	VSA-STV	15.01	10.77	-
	CTRL	24.32	18.32	-
	ACRN	24.22	19.52	-
	MCF	25.84	18.64	-
	SM-RL	26.51	20.25	-
	ACL	28.31	22.07	-
	SAP	31.15	-	-
	L-NET$^{\diamond}$	-	-	13.41
	ABLR-aw$^{\diamond}$	31.60	18.90	12.50
	ABLR-af$^{\diamond}$	34.70	19.50	13.40
	DEBUG*	35.22	22.07	14.56
	DEBUG	**41.15**	**23.45**	**16.03**

	Method	IoU@0.3	IoU@0.5	IoU@0.7
Charades-STA	VSA-RNN	-	10.50	4.32
	VSA-STV	-	16.91	5.81
	CTRL	-	23.63	8.89
	ROLE	25.26	12.12	-
	ACL	-	26.47	11.23
	SAP	-	27.42	13.36
	RWM	-	36.70	-
	SM-RL	-	24.36	11.17
	QSPN	54.70	35.60	15.80
	DEBUG*	52.16	35.89	**17.92**
	DEBUG	**54.95**	**37.39**	17.69

	Method	IoU@0.3	IoU@0.5	mIoU
ActivityNet	TGN	43.81	27.93	-
	QSPN	45.30	27.70	-
	RWM	-	36.90	-
	ABLR-af$^{\diamond}$	53.65	34.91	35.72
	ABLR-aw$^{\diamond}$	55.67	36.79	36.99
	DEBUG*	55.82	39.20	39.01
	DEBUG	**55.91**	**39.72**	**39.51**

Table 1: Performance (%) over R@1,IoU@θ and mIoU compared with the state-of-the-art NLVL models on TACoS, Charades-STA and ActivityNet Captions. $^{\diamond}$ denotes the models that are under bottom-up framework. * denotes the DEBUG which is without temporal pooling. The **best** and second best methods under each setting are marked in according formats.

Dataset	Metric	QANet-SE	DEBUG
TACoS	IoU@0.1	29.54	**41.15**
	IoU@0.3	16.75	**23.45**
	IoU@0.5	9.57	**11.72**
	mIoU	12.01	**16.03**
Charades-STA	IoU@0.3	50.81	**54.95**
	IoU@0.5	32.63	**37.39**
	IoU@0.7	16.24	**17.69**
	mIoU	33.94	**36.34**
ActivityNet	IoU@0.1	71.90	**74.26**
	IoU@0.3	53.44	**55.91**
	IoU@0.5	38.04	**39.72**
	mIoU	38.02	**39.51**

Table 2: Performance (%) over R@1,IoU@θ and mIoU compared with QANet-SE on TACoS, Charades-STA and ActivityNet Captions.

4) Others (*i.e.*, RL-based models): **RWM** (He et al., 2019), **SM-RL** (Wang et al., 2019).

Results. The results are reported in Table 1. From Table 1, we can observe that the DEBUG achieves a new state-of-the-art performance under all evaluation metrics and benchmarks. It is worth noting that DEBUG can especially improve the performance significantly in some more strict metrics (*e.g.*, 2.62% and 2.52% absolute improvement in mIoU on dataset TACoS and ActivityNet Captions, and 2.12% absolute improvement in IoU@0.7 on Charades-STA[3]), which demonstrates the effectiveness of the DEBUG.

4.4 Ablative Studies

In this section, we did extensive ablative experiments to thoroughly investigate the DEBUG.

4.4.1 Sparse vs. Dense Bottom-Up

Setting. To eliminate the influence of backbones and equally investigate the performance gain between DEBUG and the existing sparse bottom-up framework, we designed a strong baseline dubbed as **QANet-SE**. Its backbone is identical to DEBUG (Figure 3), but its head network follows the sparse bottom-up framework, *i.e.*, it predicts the start and end time points directly.

Results. The results are reported in Table 2. We can observe that DEBUG surpasses QANet-SE over all metrics and benchmarks. Especially, the performance gains are much more obvious in TACoS (*e.g.*, over 20%∼40% relative improvements in all metrics). This is because the average length of each video in TACoS is largest among all benchmarks, and the QANet-SE style (*i.e.*, sparse bottom-up) method suffers severe positive and negative samples imbalance in long videos. Instead, DEBUG can relieve this problem by utilizing much more positive training samples.

4.4.2 Importance of Each Component

We ran a number of experiments to analyze the importance of each component in DEBUG. Results are shown in Table 3 and discussed in detail next.

Classification vs. Confidence Subnet. From Table 3, we can observe that models with only a single classification or confidence subnet can get comparable performance. More precisely, the latter one is slightly better than the former. This is

[3]Since all published works had not reported their mIoU scores on Charades-STA, we only compare with them on the metric IoU@θ.

Datasets			TACoS				Charades-STA				ActivityNet			
CLS	CFD	TP	IoU@θ 0.1	IoU@θ 0.3	IoU@θ 0.5	mIoU	IoU@θ 0.3	IoU@θ 0.5	IoU@θ 0.7	mIoU	IoU@θ 0.1	IoU@θ 0.3	IoU@θ 0.5	mIoU
✓			31.67	20.30	11.61	13.47	47.39	33.92	17.12	32.03	70.54	54.09	38.64	36.97
	✓		33.74	20.34	10.97	13.83	48.52	34.68	16.40	32.32	70.93	54.94	38.85	38.48
✓	✓		35.22	22.07	11.44	14.56	52.16	35.89	**17.92**	34.04	73.34	55.82	39.20	39.01
✓		✓	37.59	22.76	11.40	15.23	51.72	34.60	16.94	34.15	73.56	55.43	39.31	38.74
	✓	✓	40.14	22.27	11.58	15.43	51.67	35.38	15.51	33.86	73.62	55.52	39.00	39.33
✓	✓	✓	**41.15**	**23.45**	**11.72**	**16.03**	**54.95**	**37.39**	17.69	**36.34**	**74.26**	**55.91**	**39.72**	**39.51**

Table 3: Performance (%) over R@1,IoU@θ and mIoU in ablative experiments of each component of DEBUG model on TACoS, Charades-STA and ActivityNet Captions. CLS: with or without the classification subnet, CFD: with or without the confidence subnet, TP: with or without the temporal pooling.

Methods	TACoS	ActivityNet
MCN	9.41	0.30
VSA-RNN	6.45	2.29
VSA-STV	3.44	0.12
CTRL	4.02	0.13
ACRN	4.09	0.13
ABLR$^\diamond$	0.14	0.02
DEBUG	**0.02**	**0.02**

Table 4: Average time (s) to localize one sentence for different methods on TACoS and ActivityNet Captions. $^\diamond$ denotes sparse bottom-up model.

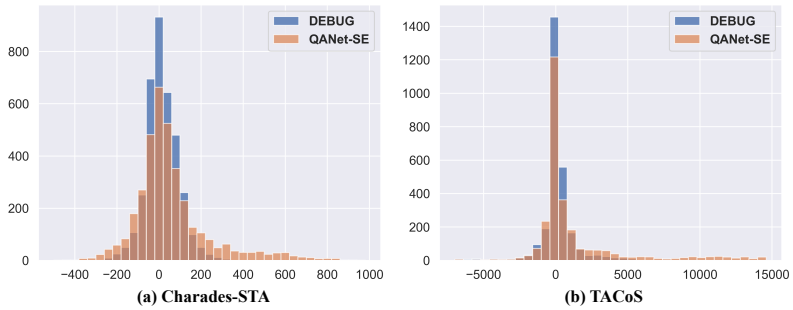

Figure 6: The number of samples across the length error (#frames difference) between predicted segment and ground truth segment on Charades-STA and TACoS.

because the confidence subnet considers the importance of each frame, instead, the classification subnet regards all foreground equally. Meanwhile, the performance of both models can be further boosted by utilizing two subnets simultaneously, which demonstrates that this multi-task design helps each subnet to focus on their own goal and both subnets benefit from sharing features (*i.e.*, one for foreground prediction, and another for "centerness" prediction).

With vs. Without Temporal Pooling. From Table 3, we can observe that the temporal pooling trick improves the performance in most of the situations, and the performance gains in TACoS are largest over all benchmarks. The main reason is that the visual appearance of each frame in TACoS is quite similar, *i.e.*, the performance based on single frame prediction is very unstable since multiple frames have similar predictions. Instead, the model with temporal pooling helps to avoid this by fusing multiple frame predictions.

4.4.3 Efficiency Analysis.

We evaluated the efficiency of DEBUG, by comparing the average run time to localize one sentence in the video. As shown in Table 4, the DEBUG significantly reduce the localization time compared to all top-down models (MCN, VSA-RNN, VSA-STV, CTRL, ACRN), and the gap is much wider in long video datasets (*e.g.*, TACoS). This meets with the notorious drawback of the top-down framework, *i.e.*, it is computation-intensive for dense sliding windows or anchors. Meanwhile, DEBUG is even slightly faster than the sparse bottom-up model (ABLR), this is because the QANet backbone only uses conv and self-attention layer instead of the time-consuming RNN in ABLR backbone. All experiments are conducted on the same hardware (an NVIDIA GTX 1080Ti).

	Methods	Charades-STA 100	Charades-STA 200	Charades-STA 300	TACoS 100	TACoS 200	TACoS 300
L	QANet-SE	54.7	73.0	81.9	**15.2**	21.8	28.1
	DEBUG	**56.5**	**75.1**	**83.2**	13.2	**22.7**	**29.8**
R	QANet-SE	50.3	69.8	81.0	**11.8**	20.1	26.7
	DEBUG	**54.6**	**75.1**	**83.2**	11.3	**20.4**	**27.5**
L&R	QANet-SE	38.5	61.5	74.4	**6.5**	12.2	16.9
	DEBUG	**46.5**	**71.1**	**81.1**	4.9	**12.8**	**19.5**
M	QANet-SE	53.9	72.7	83.1	11.6	20.5	28.2
	DEBUG	**57.3**	**75.8**	**83.7**	**13.3**	**23.2**	**30.2**

Table 5: Accuracy (%) of multiple keypoints at different thresholds on Charades-STA and TACoS. L: left point; R: right point; L&R: both left and right point; M: middle point.

4.4.4 Error Analysis.

To analyze the bottleneck of DEBUG and the existing sparse bottom-up framework, and help pave

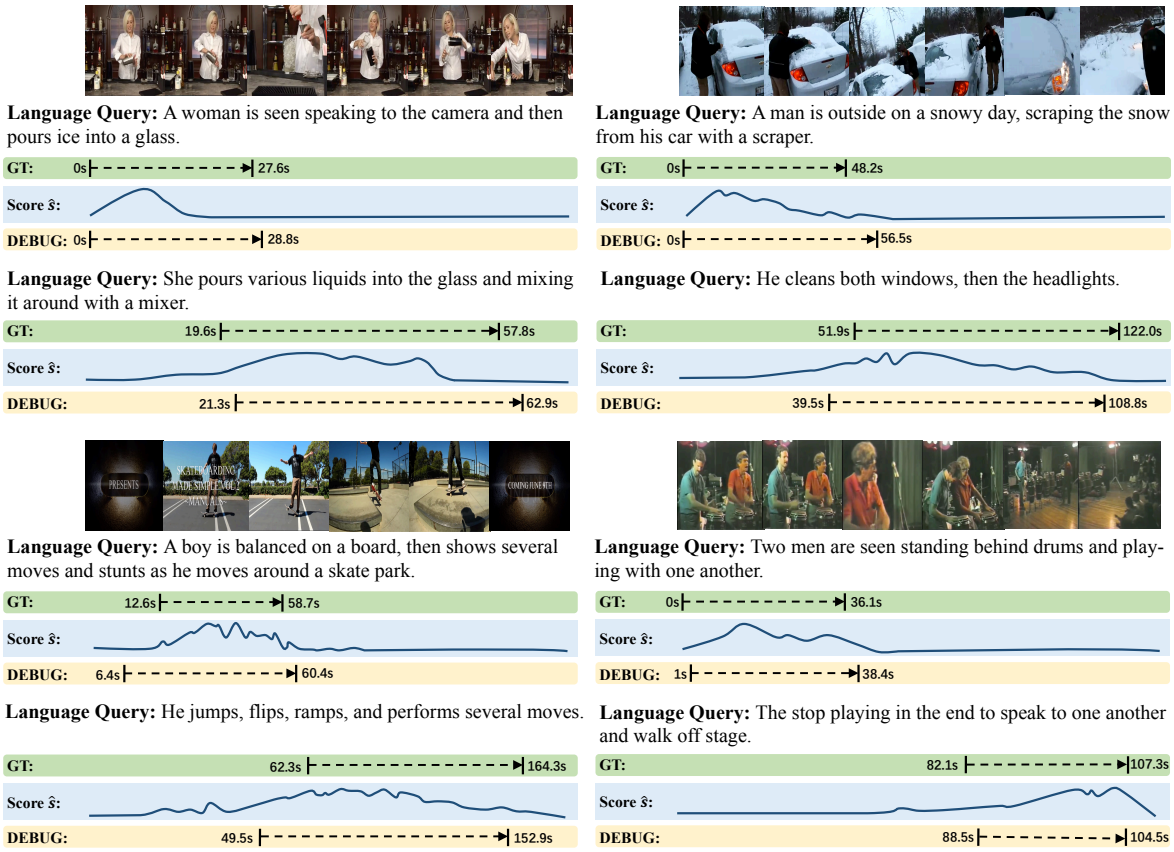

Figure 7: The qualitative results of DEBUG on the ActivityNet Captions dataset.

the way for future research on bottom-up approach for NLVL, We conducted several statistical analysis about the results of DEBUG and its sparse bottom-up counterpart QANet-SE (Section 4.4.1):

Segment Length. We compared the length error of the segments predicted by DEBUG and QANet-SE, the results are illustrated in Figure 6. We observe that QANet-SE is prone to predicting overlong segment range (*e.g.*, the samples with length error larger than 300 frames in Chardes-STA or 5,000 frames in TACoS take a large proportion.)

Keypoint Accuracy. We compared the accuracy of three keypoints of the ground truth segment: the left, right, and middle point. We regard a keypoint prediction as right if the absolute frame difference between the prediction and ground truth is smaller than a threshold. We used three thresholds (100, 200, 300) and the results are reported in Table 5. We have two observations: 1) For middle point, DEBUG always gets higher accuracy. 2) For boundary points, DEBUG only drops behind QANet-SE in TACoS at threshold 100.

Analysis. The sparse bottom-up approach (*e.g.*, QANet-SE) can get good boundary predictions even for long video datasets (*e.g.*, TACoS), which meets with the design of training a boundary classifier. But the bottleneck of this approach is that the predictions of start and end point are independent, which is prone to result in an overlong segment prediction. However, DEBUG predicts boundary from the same frame feature, which can avoid predicting overlong segments. Meanwhile, DEBUG has a confidence subnet to predict "centerness" of each frame, which helps to predict the middle point. But the bottleneck of DEBUG is the accuracy of boundary point in long video dataset.

4.4.5 Qualitative Results.

The qualitative results of DEBUG on ActivityNet Captions is illustrated in Figure 7. We can observe that DEBUG is sensitive to language query, *i.e.*, the predicted scores $\hat{s}$ are totally different when given different language queries even for the same video. Meanwhile, the score $\hat{s}$ is always a unimodal curve with its peak near the midpoint of ground truth segment, which meets with the design of DEBUG which uses "centerness" as the confidence target.

5 Conclusion

We proposed a novel dense bottom-up framework DEBUG for NLVL. It is the first bottom-up model, which surpasses all top-down models with the highest speed. Compared to the existing bottom-up models, DEBUG improve performance significantly by: 1) making full use of positive samples to alleviate the severe imbalance problem; 2) disentangling boundary detection from query-related prediction to relieve the burden of a single network; 3) predicting boundaries from same frame to avoid local optimum caused by independent predictions. Moving forward, we are going to narrow the gap between top-down and bottom-up models, and design a hybrid framework exploring both choices.

Acknowledgement

This work was supported by National Key Research and Development Program of China (SQ2018AAA010010), Zhejiang Natural Science Foundation (LR19F020002, LZ17F020001), National Natural Science Foundation of China (61976185, 61572431), the Fundamental Research Funds for the Central Universities Chinese Knowledge Center for Engineering Sciences and Technology, and Joint Research Program of ZJU & Tongdun Technology. Long Chen was supported by 2018 Zhejiang University Academic Award for Outstanding Doctoral Candidates.

References

A.Karpathy, G.Toderici, S.Shetty, T.Leung, and R.Sukthankar. 2014. Large-scale video classification with convolutional neural networks. In *CVPR*.

Peter Anderson, Xiaodong He, Chris Buehler, Damien Teney, Mark Johnson, Stephen Gould, and Lei Zhang. 2018. Bottom-up and top-down attention for image captioning and visual question answering. In *CVPR*.

Jingyuan Chen, Xinpeng Chen, Lin Ma, Zequn Jie, and Tat-Seng Chua. 2018. Temporally grounding natural sentence in video. In *EMNLP*.

Jingyuan Chen, Lin Ma, Xinpeng Chen, Zequn Jie, and Jiebo Luo. 2019a. Localizing natural language in videos. In *AAAI*.

Long Chen, Hanwang Zhang, Jun Xiao, Xiangnan He, Shiliang Pu, and Shih-Fu Chang. 2019b. Counterfactual critic multi-agent training for scene graph generation. In *ICCV*.

Long Chen, Hanwang Zhang, Jun Xiao, Liqiang Nie, Jian Shao, Wei Liu, and Tat-Seng Chua. 2017. Sca-cnn: Spatial and channel-wise attention in convolutional networks for image captioning. In *CVPR*.

Shaoxiang Chen and Yu-Gang Jiang. 2019. Semantic proposal for activity localization in videos via sentence query. In *AAAI*.

D.P.Kingma and J.Ba. 2015. Adam: A method for stochastic optimization. In *ICLR*.

D.Tran, L.Bourdev, R.Fergus, L.Torresani, and M.Paluri. 2015. Learning spatiotemporal features with 3d convolutional networks. In *ICCV*.

Kaiwen Duan, Song Bai, Lingxi Xie, Honggang Qi, Qingming Huang, and Qi Tian. 2019. Centernet: Keypoint triplets for object detection. In *arXiv*.

Jiyang Gao, Chen Sun, Zhenheng Yang, and Ram Nevatia. 2017. Tall: Temporal activity localization via language query. In *ICCV*.

Runzhou Ge, Jiyang Gao, Kan Chen, and Ram Nevatia. 2019. Mac: Mining activity concepts for language-based temporal localization. In *WACV*.

Dongliang He, Xiang Zhao, Jizhou Huang, Fu Li, Xiao Liu, and Shilei Wen. 2019. Read, watch, and move: Reinforcement learning for temporally grounding natural language descriptions in videos. In *AAAI*.

Lisa Anne Hendricks, Oliver Wang, Eli Shechtman, Josef Sivic, Trevor Darrell, and Bryan Russell. 2017. Localizing moments in video with natural language. In *ICCV*.

Lisa Anne Hendricks, Oliver Wang, Eli Shechtman, Josef Sivic, Trevor Darrell, and Bryan Russell. 2018. Localizing moments in video with temporal language. In *EMNLP*.

J.Pennington, R.Socher, and C.Manning. 2014. Glove: Global vectors for word representation. In *EMNLP*.

Ranjay Krishna, Kenji Hata, Frederic Ren, Li Fei-Fei, and Juan Carlos Niebles. 2017. Dense-captioning events in videos. In *ICCV*.

Hei Law and Jia Deng. 2018. Cornernet: Detecting objects as paired keypoints. In *ECCV*.

Meng Liu, Xiang Wang, Liqiang Nie, Xiangnan He, Baoquan Chen, and Tat-Seng Chua. 2018a. Attentive moment retrieval in videos. In *SIGIR*.

Meng Liu, Xiang Wang, Liqiang Nie, Qi Tian, Baoquan Chen, and Tat-Seng Chua. 2018b. Cross-modal moment localization in videos. In *ACM MM*.

Michaela Regneri, Marcus Rohrbach, Dominikus Wetzel, Stefan Thater, Bernt Schiele, and Manfred Pinkal. 2013. Grounding action descriptions in videos. *TACL*.

Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. 2015. Faster r-cnn: Towards real-time object detection with region proposal networks. In *NIPS*.

Zheng Shou, Junting Pan, Jonathan Chan, Kazuyuki Miyazawa, Hassan Mansour, Anthony Vetro, Xavier Giro-i Nieto, and Shih-Fu Chang. 2018. Online detection of action start in untrimmed, streaming videos. In *ECCV*.

Zhi Tian, Chunhua Shen, Hao Chen, and Tong He. 2019. Fcos: Fully convolutional one-stage object detection. In *arXiv*.

Weining Wang, Yan Huang, and Liang Wang. 2019. Language-driven temporal activity localization: A semantic matching reinforcement learning model. In *CVPR*.

Aming Wu and Yahong Han. 2018. Multi-modal circulant fusion for video-to-language and backward. In *IJCAI*.

Caiming Xiong, Victor Zhong, and Richard Socher. 2017. Dynamic coattention networks for question answering. In *ICLR*.

Caiming Xiong, Victor Zhong, and Richard Socher. 2018. Dcn+: Mixed objective and deep residual coattention for question answering. In *ICLR*.

Huijuan Xu, Kun He, L Sigal, S Sclaroff, and K Saenko. 2019. Multilevel language and vision integration for text-to-clip retrieval. In *AAAI*.

Huijuan Xu and Kate Saenko. 2016. Ask, attend and answer: Exploring question-guided spatial attention for visual question answering. In *ECCV*.

Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhudinov, Rich Zemel, and Yoshua Bengio. 2015. Show, attend and tell: Neural image caption generation with visual attention. In *ICML*.

Yunan Ye, Zhou Zhao, Yimeng Li, Long Chen, Jun Xiao, and Yueting Zhuang. 2017. Video question answering via attribute-augmented attention network learning. In *SIGIR*.

Adams Wei Yu, David Dohan, Minh-Thang Luong, Rui Zhao, Kai Chen, Mohammad Norouzi, and Quoc V Le. 2018. Qanet: Combining local convolution with global self-attention for reading comprehension. In *ICLR*.

Yitian Yuan, Tao Mei, and Wenwu Zhu. 2019. To find where you talk: Temporal sentence localization in video with attention based location regression. In *AAAI*.

Da Zhang, Xiyang Dai, Xin Wang, Yuan-Fang Wang, and Larry S Davis. 2019a. Man: Moment alignment network for natural language moment retrieval via iterative graph adjustment. In *CVPR*.

Songyang Zhang, jinsong Su, and Jiebo Luo. 2019b. Exploiting temporal relationships in video moment localization with natural language. In *ACM MM*.

Xingyi Zhou, Dequan Wang, and Philipp Krähenbühl. 2019a. Objects as points. In *arXiv*.

Xingyi Zhou, Jiacheng Zhuo, and Philipp Krähenbühl. 2019b. Bottom-up object detection by grouping extreme and center points. In *CVPR*.