Supplementary Material
Dense Regression Network for Video Grounding

In the supplementary material, we first give the training details of our DRN in Section A. Then, we illustrate the details of the video-query interaction module in Section B. Next, we detail the grounding module in Section C, followed by more qualitative results in Section D. Last, we provide more details of "centerness" in Section E. .

A. More details about our DRN

The training details of our proposed DRN are shown in Algorithm 1. With randomly initialized parameters, the location regression head often fails to produce high-quality temporal bounding box for training the IoU regression head. Thus, we propose a three-step strategy to train the proposed DRN. Specifically, in the first step, we fix the parameters of the IoU regression head and train the DRN by minimizing Equations (5) and (6). In the second step, we fix the parameters in DRN except for the IoU regression head and train the DRN by minimizing Equation(7). In the third step, we fine-tune the whole model in an end-to-end manner.

Algorithm 1 Training details of DRN.

Input: Video $V = \{I_t\}_{t=1}^{T}$; query $Q = \{w_n\}_{n=1}^{N}$

Step1: Fix the parameters of M_{iou}

1: **while** not converges **do**
2: predict matching score $\hat{m}_t$
3: predict regression offset $\hat{\boldsymbol{d}}_t$ using Equation (1)
4: update DRN by minimizing Equations (5) and (6)
5: **end while**

Step2: Fix the parameters of G, M_{match}, and M_{loc}

1: **while** not converges **do**
2: predict bounding box $\hat{\mathbf{b}}_t$ using Equation (1)
3: predict IoU between $\hat{\mathbf{b}}_t$ and ground truth
4: update DRN by minimizing Equation (7)
5: **end while**

Step3: Fine-tune G, M_{match}, M_{loc}, and M_{iou} jointly

1: **while** not converges **do**
2: predict matching score $\hat{m}_t$
3: predict bounding box $\hat{\mathbf{b}}_t$ using Equation (1)
4: predict IoU between $\hat{\mathbf{b}}_t$ and ground truth
5: update DRN by minimizing Equations (5), (6), (7)
6: **end while**

Output: Trained DRN

B. Details of video-query interaction module

The video-query interaction module consists of two parts, as shown in Figure A. The first part serves as a data preprocessor, which takes the query sentences, video frames and temporal coordinates as input and outputs the query feature and video feature (C_1). The second part is a fully convolutional network with vision-language fusion modules. It is used to fuse the video feature and query feature and construct a feature pyramid.

B.1. Video feature extractor

Instead of predicting a temporal bounding box at each frame, we exploit a more efficient way to implement our dense regression network. Specifically, we divide a video into K segments evenly. Thus, the temporal resolution of the video comes to K, which significantly reduces the computation in our model. Then, we use our model to predict a temporal bounding box w.r.t. the central frame of each segment. We set K as 32 for Charades-STA and ActivityNet Captions, and 128 for TACoS dataset. Three types of feature extractor are detailed as follows:

C3D. We use C3D [37] pre-trained on sport1M [22] to extract features. The C3D network takes 16 consecutive frames (a snippet) as input and the output of *fc*6 layer is used as a snippet-level feature vector. The feature of each segment is obtained by performing max-pooling among the snippet-level features that correspond to the segment.

VGG. We use VGG16-BN [34] pre-trained on ImageNet. VGG16-BN takes one frame as input and the output of *fc*7 layer is used as the frame-level feature. The segment feature is obtained by performing max-pooling among the frame-level features that correspond to the segment.

I3D. We use I3D [3] pre-trained on Kinetics to extract features. The I3D network takes 64 consecutive frames (a snippet) as input and outputs a snippet-level feature vector. The feature of each segment is obtained by performing max-pooling among the snippet-level features that correspond to the segment.

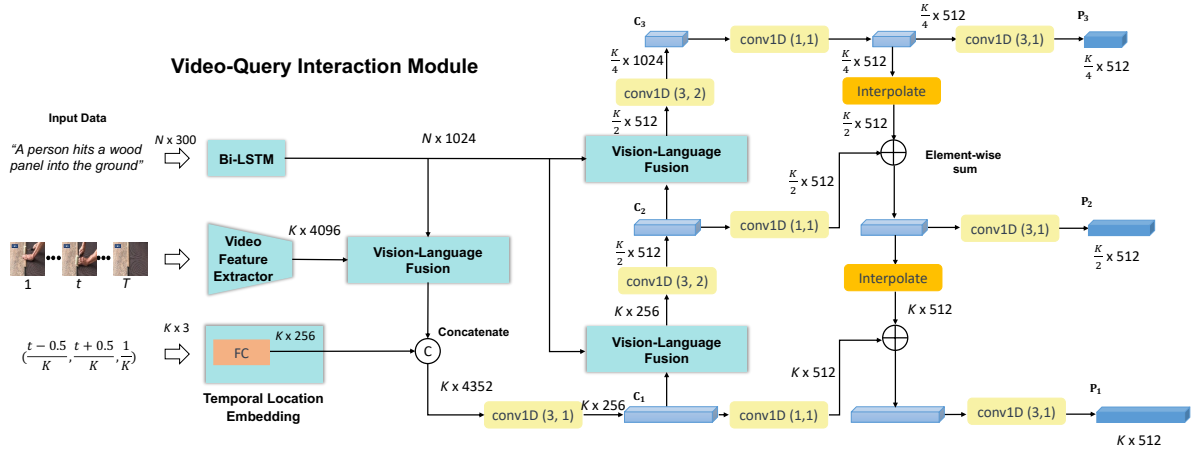

Figure A. The details of Video-Query Interaction Module. Note that "Conv1D (b, s)" denotes a 1D convolution layer with a kernel size of b and a stride of s. All the convolution layers are followed by batch normalization and ReLU.

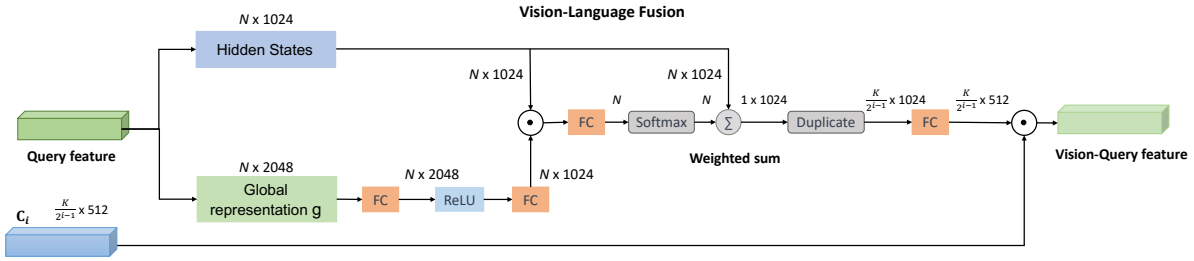

Figure B. The details of Vision-Language-Fusion Module.

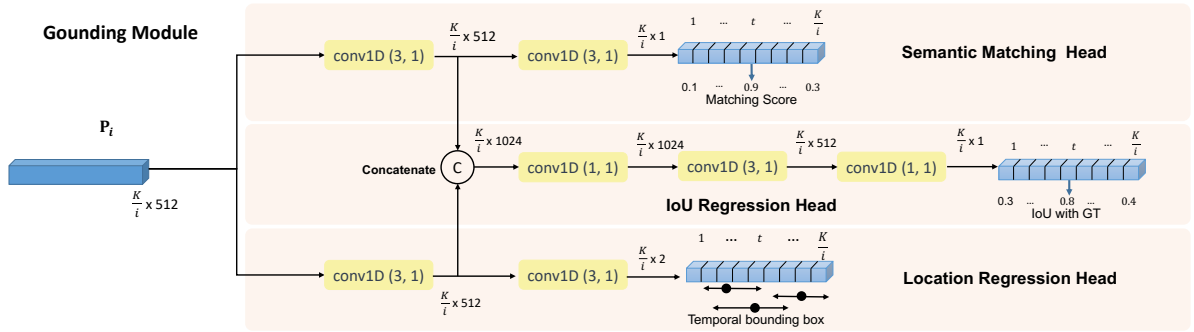

Figure C. The details of Grounding Module. The input $\mathbf{P}_i$ is from the i-th level in the feature pyramid with a temporal dimension of $\frac{K}{2^{i-1}}$.

B.2. Query feature extractor

First, each word in the input query sentence is mapped into a 300-dim vector using pre-trained GloVe word embeddings. Then, the word embeddings of the query sentence are fed into a one-layer bi-directional LSTM with 512 units. Last, the sequence of hidden states is used as query features. The hidden states of the first and the last word are concatenated, leading to the global representation g.

B.3. Location embedding

The input temporal coordinates of the k-th segment is a 3D vector, *i.e.*, $(\frac{k-0.5}{K}, \frac{k+0.5}{K}, \frac{1}{K})$. We forward it to a linear layer, leading to a 256D location embedding. The location embedding is then concatenated with the video features along the channel dimension.

B.4. Vision-Language Fusion Module

We apply the textual attention mechanism to the input query feature and obtain the attended feature. Then, the attended query features and the features from a lower level of the pyramid are fused by using element-wise multiplication. The details are shown in Figure B.

C. More details about grounding module

The grounding module involves three components, including semantic matching head, location regression head and IoU regression head. Both of the semantic matching

Figure D. Qualitative results.

head and location regression head consist of two 1D convolution layers, and IoU regression head contains three 1D convolution layers. The details are shown in Figure C.

D. More visualization examples

We show more qualitative results of the IoU regression head in Figure D. The IoU regression head helps to select the prediction that has a larger IoU with the ground truth.

E. More details about centerness

E.1. Details of centerness baseline

To compare our IoU regression with the centerness in FCOS [28], we conduct an experiment by replacing the loss function of IoU regression head with a centerness loss as in [28]. Specifically, we train the model to predict a centerness score for each location. The training target is defined as:

$$centerness* = \sqrt{\frac{min(d^*_{t,s}, d^*_{t,e})}{max(d^*_{t,s}, d^*_{t,e})}} \quad (1)$$

where $d^*_{t,s}$, $d^*_{t,e}$ are the distances between location t and the starting frame, the ending frame of ground truth boundary respectively. We follow [28] to adopt the binary cross-entropy loss as the loss function for centerness in our experiments.

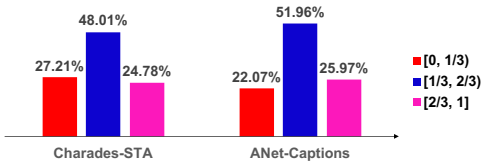

Figure E. The location distribution of the "best location" on two datasets. Here, the "best location" denotes the location that predicts the best grounding result for each video-query pair. We show the statistics of their relative locations w.r.t. the ground truth, which has been divided into three portions evenly. Here, we only focus on the locations within the ground truth since few locations fall outside of the ground truth.

E.2. Results of the centerness assumption

The centerness assumption [28] is that the location closer to the center of objects will predict a box with higher localization quality (*i.e.*, a larger IoU with the ground truth). We conduct an experiment to find out which location predicts the best box. In our experiment, we train a model using the semantic matching loss and location regression loss. For each video-query pair, we select the predicted box that has the largest IoU with the ground truth. Then, we divide the ground truth into three portions evenly and sum up the number of the locations that predicts the best box for each portion. From Figure E, more than 48% of the predictions are not predicted by the central locations of the ground truth.