BOUNDARY UNCERTAINTY IN A SINGLE-STAGE TEMPORAL ACTION LOCALIZATION NETWORK

Tingting Xie *Christos Tzelepis* *Ioannis Patras*

School of Electronic Engineering and Computer Science, Queen Mary University of London

ABSTRACT

In this paper, we address the problem of temporal action localization with a single stage neural network. In the proposed architecture we model the boundary predictions as uni-variate Gaussian distributions in order to model their uncertainties, which is the first in this area to the best of our knowledge. We use two uncertainty-aware boundary regression losses: first, the Kullback-Leibler divergence between the ground truth location of the boundary and the Gaussian modeling the prediction of the boundary and second, the expectation of the ℓ_1 loss under the same Gaussian. We show that with both uncertainty modeling approaches improve the detection performance by more than 1.5% in mAP@tIoU=0.5 and that the proposed simple one-stage network performs closely to more complex one and two stage networks.

Index Terms— Temporal action localization; uncertainty; kl divergence; One-stage network

1. INTRODUCTION

Driven by the need to process a large number of untrimmed videos generated daily by various video capturing devices, temporal action localization is drawing increasing attention from the research community [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11].

Temporal action localization typically involves first, generating video segments as candidate action proposals, and second, jointly classifying them into an action class and regressing/refining their temporal boundaries so as to better localize them in time [12, 5, 13, 7]. However, for actions in the wild, that is in unconstrained scenarios, there are large variations in how actions are performed – this makes it difficult to predict accurate boundaries. Also, unlike object boundaries, there might even be no sensible definition of what the exact temporal extent of a action is. This makes temporal boundary annotations subjective and, possibly, not consistent across different persons. Such issues are not taken into consideration by traditional regression losses used for boundary refinement (such as ℓ_1 loss [3, 12]).

To address the above issues, inspired by recent works (e.g., [14, 15]), we firstly propose to model the boundary predictions as uni-variate Gaussian distributions in temporal action localization, for which we learn their means and variances – the latter express the uncertainty about each prediction. Then, we exploit this kind of uncertainty by using two uncertainty-aware boundary regression losses. First, we use the Kullback-Leibler (KL) loss between a dirac, representing the ground truth location of the boundary and the uni-variate Guassian – this, is the cost proposed in [14] for the problem of object detection. Second, we propose to approximate the expectation of ℓ_1 loss, that is typically used as regression loss – to back-propagate the error with respect to the parameters of the Gaussian, resort to the reparametrization trick and an approximation by sampling as in [15].

Experimental evaluation of the above losses shows that the network learns to assign large variances to the samples that are predicted to be far from the ground truth boundary values. As the network converges and the predictions become more accurate, this behaviour changes and the network assigns small variances to accurate predictions. Both uncertainty-aware losses improve detection and localization performance.

The contributions of the paper are summarized as follows:

1. We propose a simple and effective one-stage network that introduces and exploits uncertainty modeling of the boundary location for temporal action localisation. To the best of our knowledge, this is the first paper that that does so in this domain.

2. For action localization we propose to use two uncertainty-aware losses: the first, based on the KL-divergence to model the difference between distributions and the second, based on the expectation of the ℓ_1 loss proposed by us.

3. We show that the uncertainty modeling improves over the adopted baseline, and that our one-stage network achieves comparable results with recent one- and two-stage networks on THUMOS'14.[1]

2. RELATED WORK

One-stage action localization detectors: Single-stage networks have been extensively used for detection [5, 16, 17, 18].

[1] Code will be made public here: https://github.com/

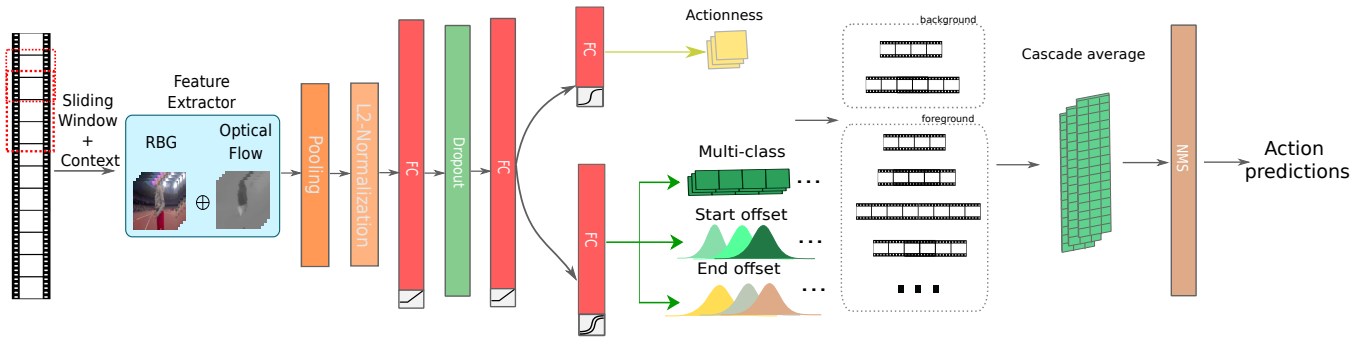


Fig. 1. Network architecture. Given an untrimmed video, sliding window proposals are generated firstly, and unit-level features are extracted by feature extractor, and then, they are passed to the network. The networks mainly constitute by two branches, one (upper branch) outputs actionness (binary) classification score to indicate if this proposal is an action; while the other one (lower branch) outputs the classification score for each class and the corresponding regression offsets (distribution) to the start and end time. During testing, temporal boundaries are adjusted in a cascaded way by feeding the refined windows back to the system for further refinement. All parameters in each cascade step are shared.

However, their performance is usually inferior than that of two-stage networks. The SSD-like detection architecture presented in [5] seems to perform well, but temporal span modeling of videos has more variations and more arbitrary compared to spatial information in objects. Thus, it is hard to use hand-designed anchor to cover them all to get the accurate boundary without explicitly modeling the temporal information, especially for actions with large duration. [16] is the first to propose an end-to-end network, but C3D feature [19] has been proved to be inferior than two-stream feature [20] used in [3, 7]. [17] exploits C3D as a feature extractor and GRU [21], a concise and elegant way to model temporal information and make predictions of the offset. However, experimental results show that GRU is not sufficient in order to learn representation for accurate localization compared to CNN-based methods. In this paper, we try to alleviate the drawback of the above methods using our one-stage network.

Uncertainty Learning in DNNs: To improve robustness and interpretability of discriminant Deep Neural Networks (DNNs), introducing and learning under uncertainty is receiving increasing attention among the research community [22, 23, 14, 24]. In this respect, two main categories of uncertainty are studied: *model uncertainty* and *data uncertainty*. Model uncertainty refers to the uncertainty of the model parameters given the training data and can be reduced by collecting additional data [22]. Data uncertainty accounts for the uncertainty in output whose primary source is the inherent noise in input data [23, 14, 24]. Despite the fact that a few methods have been proposed for dealing with data uncertainty in classification and regression problems (e.g., in segmentation [23] or object detection [14] tasks), to the best of our knowledge, this the first work that does that in the temporal action localization domain.

3. METHOD

3.1. Baseline architecture

In this work, we propose a simple single-stage network, which will serve as our baseline architecture. The proposed network draws inspiration from the standard two-stage approach that includes a proposal generation stage and a detection stage. Both of them could be considered as standard classification-regression networks that take as input a fixed-size feature representation scheme extracted from temporal clips of varying lengths. First, in the proposal generation stage a binary classification classifies the segment as being background or foreground (i.e., being one of a set of known actions). Second, in the detection stage, a coupled regression-classification scheme refines the segment boundaries and classifies it into one of the knows action classes. In this paper, we combine the two stages together into a singe-stage network that conducts end-to-end action detection and localization by having two branches to perform binary classification and multi-classification/regression separately, as shown in Fig. 1.

Specifically, for each input proposal, to partially preserve input temporal structure, we divide each input proposal into k parts and apply average pooling to each part, as in [7], to get a fixed-dimensional feature representation scheme, and then a ℓ_2-normalization layer and a fully-connected layer (along with a ReLU layer) are followed. After that, there are two branches of fully-connected layer: the first branch is only doing binary classification, which indicates if this proposal is an action or not; while the second branch is to output multi-classification scores and the refined start, and end offsets corresponding to refinements of the boundaries for each action category. Different from the lower branch shown in Fig. 1, the baseline network only predicts start offsets and end off-

sets with ℓ_1 regression loss; the distribution prediction will be described in detail in Sect. 3.4.

3.2. Classification

Before discussing the different boundary regression methods, let us define the training set with supervision as follows:

$$\mathcal{X} = \left\{ \left(\mathbf{x}^{(i)}, t_a^{(i)}, t_c^{(i)}, t_s^{(i)}, t_e^{(i)} \right) \right\}_{i=1}^{N},$$

where $\mathbf{x}^{(i)}$, $t_a^{(i)}$, $t_c^{(i)}$, $t_s^{(i)}$, and $t_e^{(i)}$ denote the feature vector, the actioness label, class label, the start, and the end offsets of the i-th training sample, respectively. t_a is binary, which indicates that a training example depicts a class or the background. t_c is a multi-class label, indicates the category a training example belongs to. Given a feature vector $\mathbf{x}$, the baseline network infers the actioness score y_a, multi-class score y_c, the start offset y_s, and the end offset y_e.

For the binary classification task, i.e., for learning the actioness score, we use the standard binary cross entropy loss. However, due to the fact that proposals that actually depict an action are far fewer than the ones that depict background, the dataset is imbalanced. To deal with this, we adopt a popular technique (see e.g. [25]), namely hard-negative mining, where we keep the ratio between positive and negative (with respect to actioness) samples fixed and equal to λ. For our experiments, we set $\lambda = \frac{1}{3}$. Then, the total binary loss is given as:

$$\mathcal{L}_{bin} = \frac{-1}{|I_p| + |I_n^\lambda|} \left(\sum_{i \in I_p} \log \left(y_a^{(i)} \right) + \sum_{i \in I_n^\lambda} \log \left(1 - y_a^{(i)} \right) \right) \tag{1}$$

where I_p and I_n^λ denote the sets of indices of the positive and chosen negative samples, respectively.

3.3. Standard multi-class classification and boundary regression

For the multi-class classification task, suppose N_c is the number of action category in the dataset, the classification loss is written by:

$$\mathcal{L}_{cls} = \frac{1}{|I_p|} \sum_{i \in I_p} \sum_{j}^{N_c} \left(-t_c^{(ij)} \log \hat{y}_c^{(ij)} \right) \tag{2}$$

where

$$\hat{y}_c^j = \frac{\exp(y_c^j)}{\sum_j \exp(y_c^j)}$$

For the regression task, i.e., for adjusting the start and the end offsets, typically ℓ_1 loss is used:

$$\mathcal{L}_{reg} = \frac{1}{|I_p|} \sum_{i \in I_p} \left(|t_s^{(i)} - y_s^{(i)}| + |t_e^{(i)} - y_e^{(i)}| \right) \tag{3}$$

3.4. Uncertainty-aware boundary regression

As discussed above, for modeling output uncertainty, we propose to model the boundary offsets as uni-variate Gaussian distributions for which their first- and second-order moments are learned by the network (see Fig. 1). That is, instead of predicting a deterministic pair of start/end boundaries, we predict a pair of uni-variate Gaussians. In the next two sections, we discuss two regression losses that exploit this kind of uncertainty; i.e., one that explicitly uses the distributions for computing the boundary regression loss and one that samples for them to approximate the expectation of ℓ_1 loss.

KL-ℓ_1 regression loss: Following similar arguments as in [14], we adopt the Kullback-Leibler divergence combined with another loss which is similar to smooth ℓ_1 loss for computing the boundary regression loss. To this end, we treat ground truth values as Dirac delta distributions, i.e., centred at the given values, in order to indicate the lack of any prior-knowledge about their uncertainty. For the sake of simplicity, if t is the ground truth value of a boundary offset, and μ, σ^2 are the mean and the variance of the corresponding network's prediction, then, for $d = t - \mu$, the following regression loss is introduced when $|d| > 1$:

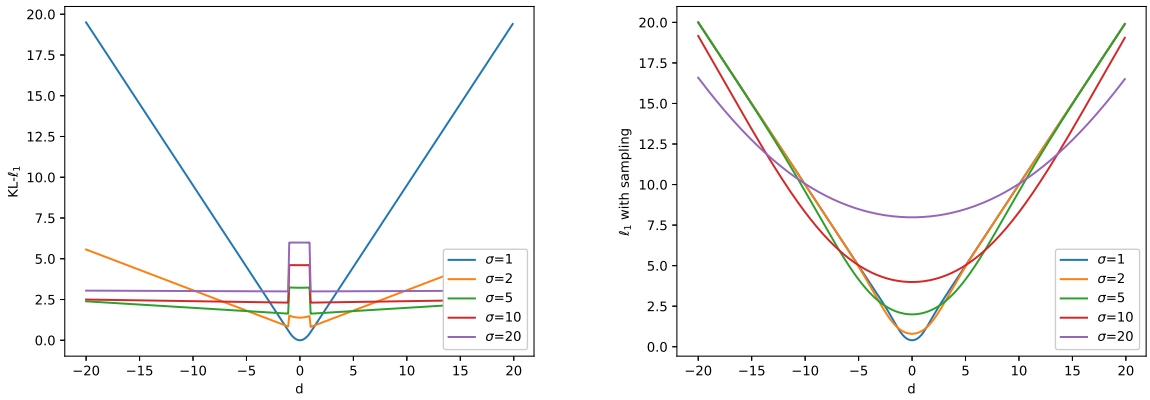
$$\mathcal{L}_{kl-\ell_1} = \frac{d^2}{2\sigma^2} + \frac{log(\sigma^2)}{2} + \frac{log(2\pi)}{2}, \tag{4}$$

and the modified smooth ℓ_1 loss when $|d| \leq 1$:

$$\mathcal{L}_{kl-\ell_1} = \frac{1}{\sigma^2} \left(|d| - \frac{1}{2} \right) + \log(\sigma) \tag{5}$$

We show the above regression loss in Fig. 2 (a). It is worth noting that for large values of d, i.e., for predicted offsets that are far from the corresponding ground truth values, loss is decreases for predictions with large variances. That is, using KL-ℓ_1 loss will force the network to predict offsets with large variances in order to converge quickly. By doing this, the network is given more freedom to discard some noisy training samples by enlarging the variances of the output. On the other hand, when the network starts to converge, i.e., when the distance between the predicted offsets and the ground truth values becomes smaller than a certain threshold, the network is trying to make the variances smaller to be accurate.

ℓ_1 regression loss with sampling: We propose an alternative uncertainty-aware boundary regression loss in order to avoid the explicit use of distributions in loss computation. In particular, at each training iteration, we sample from the predicted boundary offset distributions and compute the standard ℓ_1 loss. In this way, we approximate the expectation of ℓ_1 loss during training.



(a) KL-ℓ_1 regression loss.

(b) ℓ_1 regression loss with sampling.

Fig. 2. Uncertainty-aware boundary regression losses.

More specifically, if t is the ground truth value of a boundary offset, and μ, σ^2 are the mean and the variance of the corresponding network's prediction, at each iteration we sample from $\mathcal{N}(\mu, \sigma)$ and compute the ℓ_1 loss, i.e., the quantity $|t - y|$. However, since the sampling operation is not a well-defined differentiable operation, and thus would render back-propagation impossible, we use the well-known reparameterization trick [15]. That is, by choosing one source of randomness like the uni-variate standard Gaussian $\mathcal{N}(0, 1)$, we express the boundary offset prediction as:

$$y = \mu + \sigma\epsilon, \quad \epsilon \sim \mathcal{N}(0, 1).$$

Thus, the regression loss could be represented by (where $d = t - \mu$):

$$\mathcal{L}_{samp} = |t - \mu - \sigma\epsilon| = |d - \sigma\epsilon| \tag{6}$$

In this way, we approximate during training the expected $\mathcal{L}_{samp}$, which can be analytically be expressed as follows:

$$\mathbb{E}\left[\mathcal{L}_{samp}\right] = \mathbb{E}\left[|d - \sigma\epsilon|\right] = d\,\mathrm{erf}\left(\frac{d}{\sqrt{2\sigma^2}}\right) + \frac{\sigma \exp\left(-\frac{d^2}{\sigma^2}\right)}{\sqrt{2\pi}} \tag{7}$$

We show the expected ℓ_1 loss in Fig. 2(b). Compared with $\mathcal{L}_{kl-\ell_1}$, it doesn't have such a big tendency to predict offsets with big variances when $|d|$ is big. From the curve, it can be informed vaguely that $\mathcal{L}_{samp}$ tend to optimize d firstly, and then turn to variances. Detailed derivation can be found in the appendix.

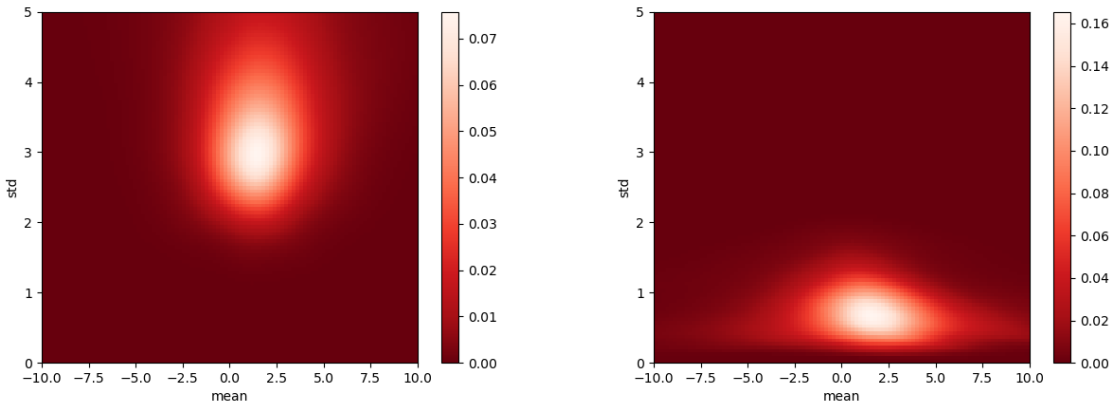
4. EXPERIMENTS

Dataset We evaluate the proposed methods on the popular THUMOS'14 [26] dataset, which contains 200 validation and 213 testing untrimmed videos, temporally annotated with 20 action classes. Following the standard practice [13, 6, 8], we train our models on the validation set and evaluate them on the testing set.

Implementation details Our baseline method is illustrated in Fig. 1. Input dimensionality of the features that feed the first FC layer is $k \cdot d$, where k is a user-defined hyper-parameter discussed in Sect. 3.1 and $d = 4096$ is the feature dimension of units (each unit consists of 16 frames) of the input proposal. The first FC layer has 1000 hidden units that feed the second FC layer, which outputs two branches. The first one predicts the actioness score (whether the input proposal depicts an action or background), and the second one predicts classification and regression scores. The output of this branch is a $C \times 3$ matrix in the baseline case, and a $C \times 6$ matrix in the case where both means and variances are predicted ($C = 20$ denotes the number of classes). During training, we used a batch size of 128, and a rate of 10^{-3}.

4.1. One- vs two-stage networks

To demonstrate the usability of our one-stage network, we compare to a similar two-stage architecture [7], for which we use proposals generated from our network. The two-stage network is constituted by a proposal generation network, and a detection network, while these two has the same classification-regression structure with ours respectively. In Table 1 we show that the proposed one-stage network achieves comparable results by involving class-agnostic along with category information in a single-stage network, with approximately half of the parameters.



(a) KL-ℓ_1 boundary regression loss.

(b) ℓ_1 regression loss with sampling.

Fig. 3. Distribution of boundary offsets means and variances using uncertainty-aware boundary regression losses.

Table 1. Two-stage and one-stage (baseline) networks performance comparison in THUMOS'14.

mAP@IoU (%)	0.3	0.4	0.5	0.6	0.7
Two-stage	49.68	44.67	36.48	24.29	13.59
One-stage	49.46	44.89	36.22	25.56	14.98

4.2. Uncertainty-Aware losses

In this section, we compare the uncertainty-aware KL-ℓ_1 and the expected ℓ_1-sampling boundary regression losses on THUMOS'14 dataset for the problem of temporal action detection. In Fig. 3, we visualize the optimal means and variances of the offsets learned after training with the above two losses. Moreover, in Table 2 we report the performance of the two networks. We observe that, compared to ℓ_1-sampling loss, KL-ℓ_1 loss encourages learning larger variances. As we discussed in Sect. 3, the network can learn more from "easy" samples, and ignore the "hard" ones by increasing their variances to enhance the detection performance, which boosts the baseline in all tIoUs by approximately 2%.

While for ℓ_1-sampling loss the variances look smaller compared to KL-ℓ_1 loss (see Fig. 3(b)), it is constrained dynamically when the ℓ_1-norm between ground truth and prediction is becoming smaller. It boosts the performance by around 1% (see Table 2) for tIoUs apart from 0.7 by constraining the uncertainty in a relatively low level.

KL-ℓ_1 vs ℓ_1-sampling regression loss Using KL-ℓ_1 boundary regression loss arrives at slightly better results than using ℓ_1-sampling loss. We argue that, due to the extreme imbalance between positive and negative proposals generated by sliding window, it's more urgent to suppress the negative noisy samples rather than boost the positive boundary prediction accuracy. While KL divergence could suppress the negative proposals by enlarging the corresponding variances; ℓ_1-sampling could give more realistic variances by boosting the regression accuracy, which leads to that.

4.3. Comparison to state-of-the-art

In Table 2 we report the experimental results of two uncertainty-aware losses compared to several related works. We note that KL-ℓ_1 loss achieves second highest performance among the single-stage methods, even comparable with current two-stage methods; and highlight that with the uncertainty estimation our result outperforms the other one-stage methods apart from [18] in a large margin by more than 5% in all tIoUs without bells and whistles. As to [18], the main stream branch uses [5] as the backbone network (they improve the backbone mAP@tIoU=0.5 from 24.6% to 31.2%, which is still not as good as ours, 37.9%), but it has two extra branches to deal with. That is, a proposal generation and a classification branch need to be trained as well, which triples the parameters of [5] to achieve the reported performance.

5. CONCLUSION

In this paper we propose an uncertainty-aware boundary regression loss for the problem of temporal action localization in videos. We model boundary offset predictions as univariate Gaussian distributions and we compute the expectation of ℓ_1 loss for improving localization. We compare with another uncertainty-aware loss that explicitly uses the predicted distributions, which we apply to the problem of temporal action localization for the first time. In the future, we intend to investigate the use of the predicted variances in the test phase in the direction of improving inference.

Table 2. Temporal action localization methods on THUMOS'14 with various tIoU thresholds.

Method	mAP@IoU (%)				
	0.3	0.4	0.5	0.6	0.7
Two-stage methods					
CDC [27]	40.1	29.4	23.3	13.1	7.9
SSN [4]	51.9	41.0	29.8	–	–
CBR [12]	50.1	41.3	31.0	19.1	9.9
Faster rcnn [13]	53.2	**48.5**	**42.8**	**33.8**	**20.8**
BSN [6]	53.5	45.0	36.9	28.4	20.0
TAD [7]	52.5	46.6	37.4	24.5	12.4
TBN [28]	53.8	47.1	39.1	29.7	20.8
BMN [8]	56.0	47.4	38.8	29.7	20.5
GTAN [29]	**57.8**	47.2	38.8	-	-
One-stage methods					
RL [1]	36.0	26.4	17.1	–	–
SSAD [5]	43.0	35.0	24.6	15.4	7.7
SAP [30]	-	-	27.7	-	-
SS-TAD [17]	45.7	-	29.2	-	9.6
Decoup-ssad [18]	**60.2**	**54.1**	**44.2**	**32.3**	**19.1**
Ours					
Baseline	49.5	44.9	36.2	25.6	15.0
$\mathbb{E}[\ell_1]$ (sampling)	50.5	45.1	37.7	26.1	14.9
KL-ℓ_1	**51.8**	**47.7**	**37.9**	**27.6**	**16.0**

6. REFERENCES

[1] Serena Yeung, Olga Russakovsky, Greg Mori, and Li Fei-Fei, "End-to-end learning of action detection from frame glimpses in videos," in *CVPR*, 2016, pp. 2678–2687.

[2] Zheng Shou, Dongang Wang, and Shih-Fu Chang, "Temporal action localization in untrimmed videos via multi-stage cnns," in *CVPR*, 2016, pp. 1049–1058.

[3] Jiyang Gao, Zhenheng Yang, Chen Sun, Kan Chen, and Ram Nevatia, "Turn tap: Temporal unit regression network for temporal action proposals," *arXiv preprint arXiv:1703.06189*, 2017.

[4] Yue Zhao, Yuanjun Xiong, Limin Wang, Zhirong Wu, Xiaoou Tang, and Dahua Lin, "Temporal action detection with structured segment networks," in *ICCV*, 2017, vol. 8.

[5] Tianwei Lin, Xu Zhao, and Zheng Shou, "Single shot temporal action detection," in *ACM Multimedia*. ACM, 2017, pp. 988–996.

[6] Tianwei Lin, Xu Zhao, Haisheng Su, Chongjing Wang, and Ming Yang, "Bsn: Boundary sensitive network for temporal action proposal generation," *ECCV*, 2018.

[7] Tingting Xie, Xiaoshan Yang, Tianzhu Zhang, Changsheng Xu, and Ioannis Patras, "Exploring feature representation and training strategies in temporal action localization," *ICIP*, 2019.

[8] Tianwei Lin, Xiao Liu, Xin Li, Errui Ding, and Shilei Wen, "Bmn: Boundary-matching network for temporal action proposal generation," *ICCV*, 2019.

[9] Runhao Zeng, Wenbing Huang, Mingkui Tan, Yu Rong, Peilin Zhao, Junzhou Huang, and Chuang Gan, "Graph convolutional networks for temporal action localization," in *ICCV*, 2019, pp. 7094–7103.

[10] Chuming Lin, Jian Li, Yabiao Wang, Ying Tai, Donghao Luo, Zhipeng Cui, Chengjie Wang, Jilin Li, Feiyue Huang, and Rongrong Ji, "Fast learning of temporal action proposal via dense boundary generator," *arXiv preprint arXiv:1911.04127*, 2019.

[11] Yuan Liu, Lin Ma, Yifeng Zhang, Wei Liu, and Shih-Fu Chang, "Multi-granularity generator for temporal action proposal," in *CVPR*, 2019, pp. 3604–3613.

[12] Jiyang Gao, Zhenheng Yang, and Ram Nevatia, "Cascaded boundary regression for temporal action detection," *arXiv preprint arXiv:1705.01180*, 2017.

[13] Yu-Wei Chao, Sudheendra Vijayanarasimhan, Bryan Seybold, David A Ross, Jia Deng, and Rahul Sukthankar, "Rethinking the faster r-cnn architecture for temporal action localization," in *CVPR*, 2018, pp. 1130–1139.

[14] Yihui He, Chenchen Zhu, Jianren Wang, Marios Savvides, and Xiangyu Zhang, "Bounding box regression with uncertainty for accurate object detection," in *CVPR*, 2019, pp. 2888–2897.

[15] Diederik P Kingma and Max Welling, "Auto-encoding variational bayes," *arXiv preprint arXiv:1312.6114*, 2013.

[16] Huijuan Xu, Abir Das, and Kate Saenko, "R-c3d: Region convolutional 3d network for temporal activity detection," in *ICCV*, 2017, vol. 6, p. 8.

[17] S Buch, V Escorcia, B Ghanem, L Fei-Fei, and JC Niebles, "End-to-end, single-stream temporal action detection in untrimmed videos," in *BMVC*, 2017.

[18] Yupan Huang, Qi Dai, and Yutong Lu, "Decoupling localization and classification in single shot temporal action detection," *arXiv preprint arXiv:1904.07442*, 2019.

[19] Quoc V Le, Will Y Zou, Serena Y Yeung, and Andrew Y Ng, "Learning hierarchical invariant spatio-temporal features for action recognition with independent subspace analysis," in *CVPR*. IEEE, 2011, pp. 3361–3368.

[20] Karen Simonyan and Andrew Zisserman, "Two-stream convolutional networks for action recognition in videos," in *NIPS*, 2014, pp. 568–576.

[21] Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio, "Learning phrase representations using rnn encoder-decoder for statistical machine translation," *arXiv preprint arXiv:1406.1078*, 2014.

[22] Yarin Gal and Zoubin Ghahramani, "Dropout as a bayesian approximation: Representing model uncertainty in deep learning," in *ICML*, 2016, pp. 1050–1059.

[23] Alex Kendall and Yarin Gal, "What uncertainties do we need in bayesian deep learning for computer vision?," in *NIPS*, 2017, pp. 5574–5584.

[24] Yichun Shi, Anil K Jain, and Nathan D Kalka, "Probabilistic face embeddings," *arXiv preprint arXiv:1904.09658*, 2019.

[25] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C Berg, "Ssd: Single shot multibox detector," in *ECCV*. Springer, 2016, pp. 21–37.

[26] Y.-G. Jiang, J. Liu, A. Roshan Zamir, G. Toderici, I. Laptev, M. Shah, and R. Sukthankar, "THUMOS challenge: Action recognition with a large number of classes," http://crcv.ucf.edu/THUMOS14/, 2014.

[27] Zheng Shou, Jonathan Chan, Alireza Zareian, Kazuyuki Miyazawa, and Shih-Fu Chang, "Cdc: convolutional-de-convolutional networks for precise temporal action localization in untrimmed videos," in *CVPR*. IEEE, 2017, pp. 1417–1426.

[28] Tao Zhang, Shan Liu, Thomas Li, and Ge Li, "Boundary information matters more: Accurate temporal action detection with temporal boundary network," in *ICASSP*. IEEE, 2019, pp. 1642–1646.

[29] Fuchen Long, Ting Yao, Zhaofan Qiu, Xinmei Tian, Jiebo Luo, and Tao Mei, "Gaussian temporal awareness networks for action localization," in *CVPR*, 2019, pp. 344–353.

[30] Jingjia Huang, Nannan Li, Tao Zhang, Ge Li, Tiejun Huang, and Wen Gao, "Sap: Self-adaptive proposal model for temporal action detection based on reinforcement learning," in *AAAI*, 2018.

7. DETAILED DERIVATION OF THE EXPECTATION OF ℓ_1 LOSS

Lemma. 1 Suppose the predicted offset is y, and the corresponding ground truth is t, the ℓ_1 loss is defined by:

$$\mathcal{L} = |t - y|$$

where $y = \mu + \sigma\epsilon, \quad \epsilon \sim \mathcal{N}(0, 1)$. The expectation of ℓ_1 loss can be analytically expressed as follows:

$$\mathbb{E}[\mathcal{L}] = d \operatorname{erf}\left(\frac{d}{\sqrt{2\sigma^2}}\right) + \frac{\sigma \exp\left(-\frac{d^2}{\sigma^2}\right)}{\sqrt{2\pi}},$$

Proof.

According to Equ. 6,

$$\mathcal{L} = |t - \mu - \sigma\epsilon| = |d - \sigma\epsilon|, \quad d = t - \mu$$

Then,

$$\mathcal{L} = \begin{cases} d - \sigma\epsilon, & if \quad d - \sigma\epsilon \geq 0 \\ \sigma\epsilon - d, & if \quad d - \sigma\epsilon < 0 \end{cases} \tag{8}$$

$$= \begin{cases} d - \sigma\epsilon, & if \quad \epsilon \leq \frac{d}{\sigma} \\ \sigma\epsilon - d, & if \quad \epsilon > \frac{d}{\sigma} \end{cases} \tag{9}$$

Thus,

$$\mathbb{E}[\mathcal{L}] = \int_{\mathbb{R}} |d - \sigma\epsilon| p(\epsilon) d\epsilon \tag{10}$$

$$= \int_{-\infty}^{d/\sigma} (d - \sigma\epsilon) p(\epsilon) d\epsilon + \int_{d/\sigma}^{+\infty} (\sigma\epsilon - d) p(\epsilon) d\epsilon \tag{11}$$

where $p(\epsilon)$ is the Probability density function of ϵ. Thus, we will know: $\int_{\mathbb{R}} p(\epsilon) d\epsilon = 1$, that is to say:

$$\int_{-\infty}^{d/\sigma} p(\epsilon) d\epsilon + \int_{d/\epsilon}^{+\infty} p(\epsilon) d\epsilon = 1 \tag{12}$$

$\mathbb{E}[\mathcal{L}]$ could be written into:

$$\mathbb{E}[\mathcal{L}] = d \int_{-\infty}^{d/\sigma} p(\epsilon) d\epsilon - \sigma \int_{-\infty}^{d/\sigma} \epsilon p(\epsilon) d\epsilon \tag{13}$$

$$+\sigma \int_{d/\sigma}^{+\infty} \epsilon p(\epsilon) d\epsilon - d \int_{d/\sigma}^{+\infty} p(\epsilon) d\epsilon \tag{14}$$

At this stage, we are going to divide $\mathbb{E}[\mathcal{L}]$ into parts and conquer one by one. Based on Equation 12, we can derive:

$$\int_{d/\epsilon}^{+\infty} p(\epsilon) d\epsilon = 1 - \int_{-\infty}^{d/\infty} p(\epsilon) d\epsilon \tag{15}$$

Then:

$$-d \int_{d/\sigma}^{+\infty} p(\epsilon) d\epsilon = -d + d \int_{-\infty}^{d/\infty} p(\epsilon) d\epsilon \tag{16}$$

As $\epsilon \sim \mathcal{N}(0, 1)$, we know $\int_{\mathbb{R}} \epsilon p(\epsilon) d\epsilon = 0$, thus:

$$\int_{-\infty}^{d/\sigma} \epsilon p(\epsilon) d\epsilon + \int_{d/\sigma}^{+\infty} \epsilon p(\epsilon) d\epsilon = 0 \tag{17}$$

Thus:

$$\sigma \int_{d/\sigma}^{+\infty} \epsilon p(\epsilon) d\epsilon = -\sigma \int_{-\infty}^{d/\sigma} \epsilon p(\epsilon) d\epsilon \tag{18}$$

Finally, based on Equation 13, 17 and 18, we can derive $\mathbb{E}[\mathcal{L}]$ into:

$$\mathbb{E}[\mathcal{L}] = 2d \int_{-\infty}^{d/\sigma} p(\epsilon) d\epsilon - d - 2\sigma \int_{-\infty}^{d/\sigma} \epsilon p(\epsilon) d\epsilon \tag{19}$$

$$= 2d\Phi(d/\sigma) - d + 2\sigma \frac{\exp(-(d/\sigma)^2)}{2\sqrt{2\pi}} \tag{20}$$

$$= d\left[2\Phi(d/\sigma) - 1\right] + \sigma \frac{\exp(-(d/\sigma)^2)}{\sqrt{2\pi}} \tag{21}$$

$$= d\left[2\frac{1}{2}(1 + \operatorname{erf}(d/\sqrt{2\sigma^2})) - 1\right] + \sigma \frac{exp(-(d/\sigma)^2)}{\sqrt{2\pi}} \tag{22}$$

$$= d \operatorname{erf}(d/\sqrt{2\sigma^2})) + \sigma \frac{exp(-(d/\sigma)^2)}{\sqrt{2\pi}} \tag{23}$$