Programming Assignment 2: PoseRAC

Bingnan Li
2020533092
libn@shanghaitech.edu.cn

Abstract

In this assignment, I explored the state-of-the-art repeat action counting algorithm **PoseRAC** *and tried to reproduce the results in the original paper. Then I proved the necessity of Transformer encoder by canceling the encoder part and using Fully-Connected Layer only. Moreover, I also tried to explore the relationship between counting performance and the number of encoder layers together with number of heads in the multi-head attention module. The results show that the performance of PoseRAC barely improves when the number of encoder layers increases from 1 to 8, but the model will crush when the number of encoder layers is larger than 8 in my training setting. Besides, the performance of PoseRAC is not sensitive to the number of heads in the multi-head attention module. Finally, I replaced the triplet margin loss with* **contrastive loss** *and* **circle loss***, the results show that circle loss significantly boosts the model within 20 epoch, the evaluation metrics MAE and OBO improved from 0.2540 and 0.5395 to 0.2083 and 0.6053.*

1. Introduction

PoseRAC is the state-of-the-art repeat action counting algorithm proposed by [**?**]. This model achieves tremendous success in the repeat action counting task and improves the performance of the previous state-of-the-art model by a large margin. The novelty of PoseRAC is the new annotation method. The traditional method will annotate the start and the end frame of an action, models are forced the regress the locations or indices of an action. However, PoseRAC turns the annotation into two salient frames which indicates the most representative points of an action. Then PoseRAC utilizes a keypoint extractor to transform the salient frames into a series of keypoint with 3D coordinates. This operation enormously reduces the amount of data need to process and improves the effectiveness of information because it only uses transformer encoder layer to get the embedding information and a single FC layer to classify the embedding.

Model	MAE	OBO
PoseRAC	0.2540	0.5395
w/o Transformer	0.9928	0.0263

Table 1. The performance of PoseRAC with and without transformer encoder.

2. Necessity of Transformer Encoder

In this section, I will prove the necessity of Transformer encoder by canceling the encoder part and using Fully-Connected Layer only. By the original implementation, the input of the model is a series of keypoint with 3D coordinates, the output of the model is the score of different salient frame of different actions. The keypoint data is fed into a transformer encoder layer and a fully-connected layer to get the final output. Since the transformer does not change the dimension of the input, I can directly cancel the transformer encoder layer and use a fully-connected layer to get the final output to explor how essential transformer is to the model.

The comparison of the performance of PoseRAC with and without transformer encoder is shown in Table 1. This result shows that the performance of PoseRAC is significantly degraded when the transformer encoder is canceled. Even the training loss (Binary Cross Entropy) drops to the same level of that of the original model, the evaluation metrics MAE and OBO are still much worse than the original model.

This experiment proves that directly classifying the keypoint data with 3D coordinates (normalized to $[0, 1]$) cannot fit the test data distribution and the transformer encoder embeds the keypoint information into a feature space that are highly separable and truly represents the common features for the same action.

Training Setting:

- lr: 0.00025
- α: 0.01
- epoch: 20

head/layer	1	2	4	6	8	≥ 10
1	0.267	0.256	0.245	0.258	0.252	-
3	0.294	0.272	0.238	0.248	0.264	-
9	0.275	0.267	0.255	0.244	0.246	-
11	0.280	0.242	0.229	0.232	0.256	-
33	0.287	0.262	0.253	0.275	0.225	-

Table 2. MAE of different parameter combinations

head/layer	1	2	4	6	8	≥ 10
1	0.467	0.526	0.507	0.507	0.533	-
3	0.467	0.507	0.539	0.533	0.480	-
9	0.487	0.474	0.513	0.474	0.533	-
11	0.474	0.526	0.546	0.520	0.461	-
33	0.461	0.487	0.526	0.474	0.513	-

Table 3. OBO of different parameter combinations

3. Impact on the Number of Encoder Layers and Number of Heads

In this section, I will explore the relationship between counting performance and the number of encoder layers together with number of heads in the multi-head attention module. In the original implementation, the number of encoder layers is 6 and the number of heads is 9. Intuitively, the more encoder layers and heads, the more information the model can capture. So, I set the number of encoder layers to be (1, 2, 4, 6, 8, 10, 12) and the number of heads to be (1, 3, 9, 11, 33).

Moreover, in order to guarantee the convergence of models, I set the max training epoch from 20 to 50 and test the models which leads to the lowest validation loss.

The experiment results are shown in Table 2 and TAble 3 and the corresponding curve is shown in Figure 1 and Figure 2.

From the experiment results, I figured out that the performance of PoseRAC can not be further improved either by increasing num of heads or num of encoder layers and the model will even crush if the num of layers increases larger than 10.

Training Setting

- lr: 0.00025
- α: 0.01
- epoch: 50

4. Impact on Training Loss

In this section, I will explore the effect of training loss on the performance of models. Given that in Section 3, I discovered that the number of heads and the number of layers can not boost the model even further, which means the transformer encoder is capable enough to embed the keypoint information even with 1 layer and 1 heads, so I try to modify the training loss and check its effect.

In the original paper, the training loss consists of two parts: TripletMarinLoss and BinaryCrossEntropyLoss. The former loss is used to push the embedding of different classes far enough while keeping the pair in the same class as closely as possible and the latter loss is used to supervise the classification process. Since the BinaryCrossEntropyLoss is widely used in classification tasks and barely has improved modifications, our focus will be the modifications of TripletMarinLoss.

In detail, I replaced TripletMarinLoss with ContrastiveLoss [**?**] and CircleLoss [**?**], the results are shown in Table 4. The results show that CircleLoss tremendously improved the model in both MAE and OBO while Con-

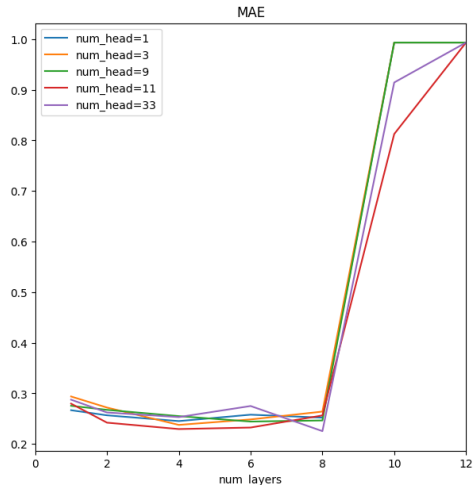

Figure 1. MAE curve of different number of encoder layers and number of heads.

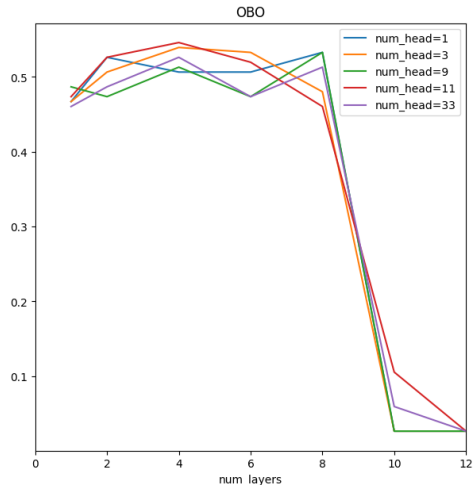

Figure 2. OBO curve of different number of encoder layers and number of heads.

Loss/Metrics	MAE	OBO
ContrastiveLoss	0.2550	0.4737
TripletMarinLoss	0.2540	0.5395
CircleLoss	**0.2083**	**0.6053**

Table 4. Impact of Losses on performance

trastiveLoss get a slight decrease in MAE but an enormous decrease in OBO compared with TripletMarinLoss.

Training Setting

- lr: 0.00025
- α: 0.01
- epoch: 20
- heads: 9
- layers: 6
- circle_loss:
  - m: 0.4
  - γ:80
- contrastive_loss:
  - pos_margin=0
  - neg_margin=1

5. Shortcut of PoseRAC

Even PoseRAC has achieved success in PAC, but it still has many shortcut in model design:

PoseRAC output is $\#\{class\}$-length vector, in the original setting, 0 represents salient frame 2 and 1 represents salient frame 1, but the annotation for salient 2 is an all-zero vector with size $\#\{class\}$. In other words, PoseRAC can not discriminate the salient frames of different classes since no matter what classes the salient 2 belongs to, the corresponding annotate is always an all-zero vector. The way the author addresses this is a kind of brute force: he traverses all action classes and performs the counting process and takes the highest class result as the final result. But if the number of classes increases from 8 (original setting) to 100 or even 1000, the computing performance may drop and can not even run in real time.

References

[1] Ziyu Yao, Xuxin Cheng, and Yuexian Zou. Poserac: Pose saliency transformer for repetitive action counting. *arXiv preprint arXiv:2303.08450*, 2023.