Loss

Qingfu Wan

strawberryfgalois@gmail.com

Abstract

*This memo features brief explanation about different losses in training **I1** of integral regression in this repo, namely **(1) Adaptive I1 (2) Adaptive H1 (3) Manual**.*

1. Introduction

Rule No.1 : We would like to eradicate the intervention of human labor *e.g.* tuning hyper parameters insofar as it's attainable.

2. Adaptive H1

With the accretion of depth dimension d_2, the contribution ratio between *2D heatmap euclidean loss* and *3D heatmap euclidean loss* changes. Fig 1 shows usage of *Adaptive H1*. The four bottom blobs are

1 *flattend 2D heatmap prediction*

2 *flattend 2D heatmap ground truth*

3 *flattend 3D heatmap prediction*

4 *flattend 3D heatmap ground truth*

3. Adaptive I1

The only difference between *Adaptive H1* and *Adaptive I1* is that *Adaptive I1* takes into account the integral of *2D joint prediction* (dimension x, y) and *joint depth prediction* (dimension z). This is clearly manifest in Fig 2.

Therein $pred\_joint\_2d\_s2\_int$ is *2D integral* from **3D heatmap prediction**, $pred\_joint\_2d\_s1\_int$ is *2D integral* from ***2D* heatmap prediction**. As to annotation, $crop\_gt\_joint\_2d$ is 2D ground truth, while $annot\_depth$ is ground truth of joint depth (dimension z).

4. Manual

Manual tuning is introduced when at a depth dimension, say *16*, making use of *Adaptive I1* cannot compete with *H1*. It cannot be ascertained that *Adaptive I1* can always achieve best result. And so it's high time to train it in old-fashioned way. Fig 3 displays the *euclidean loss layer* of *2d heatmap* and *3d heatmap*. We see the ratio is $1.0 : 0.3$.

Speaking of *loss weight*, below lists some related work [1] [2] [3].

One last thing, the performance of $d_2 = 32$, in my view can be further improved by *Manual*. And the performance gap between $d_2 = 32$ and $d_2 = 64$ is perhaps due, in large part to the same constant *heatmap2 init std*: 0.002 from $d_2 = 32 \rightarrow d_2 = 64$. In contrast, the H1 repo changes *heatmap2 init std* $0.001 \rightarrow 0.0003$ while $d_2 = 32 \rightarrow d_2 = 64$. In a similar manner, *heatmap2 init std* can be changed $d_2 = 0.002 \rightarrow d_2 = 0.0006$ lifting $d_2 = 32 \rightarrow d_2 = 64$, which may bring advancement.

Note I have only tried *gaussian std* initialization due to some historical reasons. *MSRA* or *Xavier* would probably work also.

The reason I did not start with $d_1 = 1$, $d_2 = 64$ is because, I found that the *MPJPE* on training set is already too high to reveal any evidence of decreasing. Did not pan out. **JUST PERSONAL EXPERIENCE**. **DOES NOT MEAN ANYTHING**.

5. Conclusion

Good luck!

References

[1] Z. Chen, V. Badrinarayanan, C.-Y. Lee, and A. Rabinovich. Gradnorm: Gradient normalization for adaptive loss balancing in deep multitask networks. *arXiv preprint arXiv:1711.02257*, 2017. 1

[2] M. Guo, A. Haque, D.-A. Huang, S. Yeung, and L. Fei-Fei. Dynamic task prioritization for multitask learning. In *European Conference on Computer Vision*, pages 282–299. Springer, 2018. 1

[3] A. W. Yu, L. Huang, Q. Lin, R. Salakhutdinov, and J. Carbonell. Block-normalized gradient method: An empirical study for training deep neural network. 2018. 1

```
layer {
  bottom: "heatmap"
  top: "heatmap_flatten"
  name: "heatmap_flatten"
  type: "Flatten"
}

layer {
  bottom: "heatmap2"
  top: "heatmap2_flatten"
  name: "heatmap2_flatten"
  type: "Flatten"
}

layer {
  bottom: "label_2dhm"
  top: "label_2dhm_flatten"
  name: "label_2dhm_flatten"
  type: "Flatten"
}

layer {
  bottom: "label_3dhm"
  top: "label_3dhm_flatten"
  name: "label_3dhm_flatten"
  type: "Flatten"
}

#======adaptive loss
layer {
  name: "ada_loss"
  type: "AdaptiveWeightEucLoss"
  top: "ada_loss"
  bottom: "heatmap_flatten"
  bottom: "label_2dhm_flatten"
  bottom: "heatmap2_flatten"
  bottom: "label_3dhm_flatten"
  loss_weight: 1.0
}
```

Figure 1. Adaptive H1

```

  bottom: "pred_joint_2d_s2_int"
  bottom: "crop_gt_joint_2d"

  bottom: "pred_depth_s2_int"
  bottom: "annot_depth"

  bottom: "pred_joint_2d_s1_int"
  bottom: "crop_gt_joint_2d"

  loss_weight: 1.0
}
```

Figure 2. Adaptive I1

```

layer {
  name: "loss_2dhm"
  type: "EuclideanLoss"
  bottom: "heatmap"
  top: "loss_2dhm"
  bottom: "label_2dhm"
  loss_weight: 1.000

}
```

```

layer {
  name: "loss_3dhm"
  type: "EuclideanLoss"
  bottom: "heatmap2"
  top: "loss_3dhm"
  bottom: "label_3dhm"
  loss_weight: 0.3
  #loss_weight: 0.3 (d=4)
  #loss_weight: 0.1 (d=2)
  #loss_weight: 1.0 (d=1)

}
```

Figure 3. Manual