变长用卷积

1. external sign-language LM? We have, PHOENIX-2014. See Niu Zhe's ECCV 20. Now use it after beam search, can make wer 0.4-0.5 lower

2. Transformer? Sign Language Transformers--CVPR 2020

3. online/streamable, Monotonic SLR? FCN--ECCV 2020,

4. Can tricks on ASR be used on SLR? e.g., mWER loss, focal loss, weighted WER? mWER+RL was used in ICIP paper.

5. SE block for long-term/global information? Try to use this in FCN, Two kinds of SE, global pooling along time axis or feature axis.

Actually, both of the two kinds of SE block can be combined maybe before the input of transformer (serve as embedding, like speech-transformer)

6. gated conv/attention? focus on mouth and hand. See the paper in Interspeech 2020 (Temporal Deformable Convolution)

7. multi-lingual

8. Can transformer be used for visual feature extraction?

9. from paper 14, how to add temporal fusion? Transformer has the sequence modeling ability.

10. Use TCN to model temporal semantics, dilated conv/dense for long term

11. Data imbalance--long tail. Maybe try focal loss

12. GAN. SLP, DualLip, NMT -- NAACL paper, It seems that GAN doesn't work well for NLP, except combining with RL.

13. mask conv. in gan for tts. work.

14. slow-fast. It shows that slow-only is better than fast-only.

16. hiearchical SE (diffrent window size), like Attention-FA module in COOT for temporal attention, also to get sentence-level feature

17. more complete dual learning (SLP), other way of temporal modeling (BLSTM, etc.), sentence-level feature can also use se (now is GAP)

Niu Zhe: whether model focus on useful information? many repeats in training set but only a few in test set. Better representations

18. VideoMix. Intra-video: temporal videomix within a local window. The loss doesn't change. BAD

Or inter-video: swap a window of video2 and video1, then loss should be average. BAD

19. Aligner in E2E A TTS, length loss, soft DTW. Maybe can give up ctc using length predictor.

Aligner is used for text-to-speech, i.e., short to long. I haven't found a way to predict length from long to short. Maybe we can only use rnn...

Two path, CTC + aligner path, to predict the ratio of each frame to the gloss. And the end of 'gloss' is the sum of '1'

20. In SE we use sigmoid. In attention we use softmax. Dilated conv in SE\_conv?

21. use lm during beam search. Now it is after beam search.

22. Consistency between frame-level, 1st-level, 2nd-level representations.

In first several epochs, we 'pretrain' the model to learn the 'consistency', i.e., self-supervised,

then we use gt gloss to train the model, i.e, supervised. Also momentum-average. Fix gloss embedding GRU and update frame-level GRU?

23. CNN+transformer. Not better than TCN

24. Fully end-to-end. use RNN to decode. (exposure bias from Niu Zhe, maybe schedule sampling?)

25. track hands? object tracking

26. deformable conv, 2d, 1d

28. transformer(SUB<DEL), TCN(SUB>DEL), maybe GFE in FCN works for transformer? because transformer use random\_drop only.

29. another ctc loss/decode from 1st level

30. dual\_v2, use soft\_DTW as consistency loss fun instead of cosine

32. TCN+TF, temp\_scale. Not better than TCN

33. GLU (work), comparison between glu and se\_mix, swish. To show GLU really works, we don't use the half to gate the other half, just relu.

Try residual (sigmoid may suffer from gradient vanishing)

34. store frame-level features

35. CSAN, restrict self-attention with a local area

36. se before max pooling

37. Try to see if fix dropping works, BAD. stride transformer?

38. Relative postional encoding (Niu Zhe's scripts, Transformer-XL, Conformer),

Gaussian Kernel (aligner in E2E Adversarial TTS, should still use absolute pe after embedding? (yes) Clamp? Combine with learnable rpe? (best)),

seperate heads, local heads and global heads.

gate to fusion.

swish in FFN.

39. temporal attention (key frame)

40. use conv instead of max pooling to downsample

41. length penalty when decoding

----------------------------------------------After ICCV 2021 submission----------------------------------------

42. full transformer, transformer for visual module. Use keypoint to crop patch.

43. How to use pose?

Use pose keypoints to crop feature maps?(STMC)

Motion of keypoints.

Pretrain.

Pose and deformable convnets:

Keypoints guide offset. Supervise central kernel of first dcn\_v1 layer is better; supervise all 9 masks of first dcn\_v2 layer is better. 5 dcn layers in total. Note: There would be difference if we retrain models since the coords are normalized with (H-1) but we compute loss with H.

Heatmap guide mask. Consistent on dev and test.

Not converge? Larger factor? (0.2 is better) L2 loss? Better than L1

Replace mask? (detach or not Bad): Supervision should be better than replace since supervision can train the parameters before adding the supervision, but replace cannot.

supervise offsets and mask simultaneously? Bad since offsets are not learned well.

Combine self-learned mask with prior heatmap? Directly dot-product? Bad Gaussian?

Mask without offsets? Better than vanilla DCN

Self-generated heatmaps (like STMC)?

Downsample considering receptive field size? Also effective receptive size (gaussian weights)

Attentional pooling, i.e. not average pooling at the end.

Supervise the 1st attention block/all blocks/one block for each resolution/all blocks on a single resolution?

From the perspective of knowledge distillation?

Activate after several epochs?

Concatenate pose heatmap with RGB images? (evolve attention). Worse

Coordinates-centered Gaussian heatmap? Slightly better

Hyperparameter “pose\_f” default is 1.0

Divide to 3 patches (head and 2 hands) Worse. Divided by keypoint coordinates?

Temporal deformable transformer. For self-attention only, maybe we don’t need q-k interactions, (Deformable DETR)

44. Pay more attention on hands. Pose estimation -> 2 heatmaps (softmax) -> mask. good before 1st/2nd pooling layer

45. Signer adaptation.

46. Crop position should be “down”.

47. temporal localization

48. Multilingual by knowledge distilling/transfer learning

49. coarse to fine

50. Make D adaptive, D depends on interval? TCN to predict D

51. consistency: distribution(VAC), feature, semantics

52. mesh/point cloud

53. Dual. Cosine between features after transformer (gloss features) and visual features.

54. Combine with Niu Zhe’s methods. (SGS (bad), fine-grained labeling)

55. sign embedding (nn.Embedding)

56. heatmap guided frame dropping, select one from two consecutive BAD, seems overfitting

57. Region feature based Visual-Language-Pretraining (VLP)

58. entropy CTC, maximum entropy loss Slow and bad

59. Fingerspelling. Track hands? Bad, this task needs only one hand, all other information are irrelevant

60. Put DTCN before SAN (now is after SAN before FFN) Good. But not novel, QANet

61. Multi-modality BERT, e.g., VideoBERT, but need large amount of pretraining data.

62. Off-the-shelf segmentation model, then separate signer and background?

63. Multi-modality interaction. Also consider 49. (use entropy of attention map to see whether transformer converge?) gloss embedding table. It is just a sequence labeling task...

64. Treat video and gloss as different “views” of the same data. InfoNCE loss in “Cross-Modal Contrastive Learning for Text-to-Image Generation”.

65. Hierarchical transformer? Gloss-level, sentence-level. Intra-gloss, inter-gloss (TCN\*2+TF\*2) Better on test set with gaussian bias, but if using gaussian bias it is not inter-gloss.

66. visual module + CTC. Bad. Sequential module is important.

67. Use ISLR to pretrain, because large amount? CSLR as downstream task.

68. Coordinate attention (ratio=channel?), combine with pose. CBAM? CBAM is good.

69. Domain adaptation. Isolated v.s. continuous. GSL v.s. CSL. Transfer GSL visual module to CSL?

70. Few-shot learning

71. Retrieval-enhanced. During training process, construct a gloss (including blank) frame embedding dictionary. Before send into FC-layer, firstly look up the dictionary using kNN. Then compute the auxiliary cross-entropy loss. (GFE in FCN).

----------------------------------------------After AAAI 2022 submission----------------------------------------

1. Try to combine pose supervision with learnable window-size. If succeed, this one can at least be used as an extension for future journal submission.
2. Consider self-/semi-supervised learning. Use pose heatmap to pretrain visual module (the last 2 layers may not be trained). Use distribution consistency (VAC) to pretrain sequential module and left visual module layers (SimSiam-like).

If self-/semi-supervising doesn’t work, then one month before the ddl of CVPR 2022:

Patience=3

1. Make the conclusion of CBAM (modified) and DCN consistent: supervising the 1st block is the best. Normalize with the number of blocks.
2. Upsample spatial attention mask then supervise.
3. Try attentional pooling for VGG11. Then also use pose heatmap to supervise it.
4. From the perspective of video detection. Alignment issue: gloss probability.
5. Relation Network.

Do self-/semi- until October. If not work, go back to spatial attention (cas\_bef\_san backbone).

If one-stage rejected:

Add spatial attention to our LCSA and submit to CVPR

Else:

If self-/semi-supervised learning does not work:

Focus on spatial attention and submit to CVPR

Else:

Directly submit to CVPR

If AAAI review is bad:

Withdraw and consider adding spatial attention, then submit to CVPR

9.11-9.30: self-/semi-supervised learning

9.30-10.15: SS work then continue; otherwise spatial attention on LCSA/cas\_bef\_san

10.15-11.3: If one-stage reject, then spatial attention on LCSA; otherwise, same as before 10.15

11.3-11.16: If reviews are bad, polish, spatial attention on LCSA

PQE: late Nov., late Dec. but before Christmas, early Jan.

How to generate heatmap groundtruth?

Use heatmap as indicator of importance of each pixel:

Exp(-distance), then argmax

Use heatmap as attention distribution:

2D Gaussian distribution for each keypoint, if need n-in-1 heatmap, should use average of different keypoint distribution.

How to use pose?

Pretrain(distill):

BSL-1K--Scaling up co-articulated sign language recognition using mouthing cues\_ECCV20

Direct to sign:

BSL-1K--Scaling up co-articulated sign language recognition using mouthing cues\_ECCV20

Pose-based Sign Language Recognition using GCN and BERT\_WACVW21

Neural Sign Language Translation based on Human Keypoint Estimation\_18

Crop:

USTC-Spatial-Temporal Multi-Cue Network for CSLR (HRNet)

Multimodal Sign Language Recognition via Temporal Deformable Convolutional Sequence Learning

Multi-task:

USTC-Spatial-Temporal Multi-Cue Network for CSLR (HRNet)

Fingerspelling Detection in American Sign Language\_CVPR21

Multi-modal:

Skeleton Aware Multi-modal Sign Language Recognition\_CVRPW21 (HRNet)

Analysis on the idea of basis:

Consider heatmap size H\*W, kernel size k\*k.

Vanilla DCN: need to predict 2k^2HW items for each feature map.

We can use PCA to find principal components. Suppose we select top-n singular values, then for each feature map, we only need to predict n coefficients. Using a weight-average on the n basis, we can get the offsets of this feature map.

1. How to do PCA? Consider eigen-face, we must firstly have some face images in the training set. Then we compute the covariance matrix, and solve the singular values. For our problem, the “image” is “offset”. This means that we still need to generate pixel-specific offsets for each feature map, i.e., the cost is 2k^2HW. Thus the training cost cannot be saved, only the cost of inference is saved.

2. How to predict the n coefficients? The point is that we can only get the basis after training. Suppose that we have some layers to predict n coefficients, but during training we cannot train them. So, this may not work for inference.

3. Directly predict 2k^2HW basis instead of using PCA. How to decide n?

4. Directly predict 2k^2 basis. Then we need to predict n coefficients for each pixel positions. Cost becomes 2k^2HWn.

Summary: I think the key point is whether we need pixel-specific offsets, i.e. different offsets for different pixels. If so, probably we cannot avoid to predict 2k^2HW items. If not, maybe we can divide feature map into patches, and pixels in each patch share the same offsets (maybe not reasonable):

1. The dominant component is HW. To reduce this number, maybe we can consider to reduce HW, e.g., downsample and predict offsets on the downsampled feature map. Then upsample by interpolation.
2. How to predict N coefficients. Firstly, the coefficients are for the whole feature map. If we predict them using some pooling mechanism, I think it may damage the performance (lose spatial information).