AV-HuBERT with Multi-Resolution Attention

Stanford CS224N Custom Project

Jacob Donley and Paul Calamia
Department of Computer Science
Stanford University
{jdonley, pcalamia}@stanford.edu

Abstract

In the context of self-supervised representation learning for audio-visual speech understanding, we hypothesize that the value of historical context is non-uniform like it is in language, varying in relevance to the prediction of the next output segment in a sequential model architecture, and that attention weights would benefit from similar non-uniformity. To this end, we have augmented the multi-modal (speech and video) AV-HuBERT architecture with adaptive context pooling (ACP) to incorporate a form of multi-resolution attention in the decoder. ACP previously has been shown to improve the performance of machine learning models for neural machine translation, language modeling, and image classification, and should be applicable here. Through this modification, we expected to reduce the word error rate (WER) on a downstream lip-reading task using the Lip Reading Sentences 3 (LRS3) dataset as compared to the original AV-HuBERT implementation which uses standard self-attention in its transformer layers. The results of our augmentation suggest that ACP affords only a small benefit to AV-HuBERT in terms of WER during inference of speech from mouth-cropped videos of talkers, but increases the model training time by a factor between 5 and 10. Surprisingly, performance metrics for training the model, such as sub-word accuracy and perplexity, were not well correlated with WER at inference time.

1 Key Information to include

- Mentor: Yuan Gao
- External Collaborators: None
- Sharing project: No
- Code repo: `https://github.com/jdonley/Adaptive-Context-AV-HuBERT`

2 Introduction

Speech understanding is often considered as an audio-only problem, but the high correlation between speech waveforms and lip movements can be leveraged in multi-modal approaches to enhance performance, particularly in scenes with high acoustic noise levels. [1] Various methods have been described which utilize audio-visual (AV) information from videos of talkers for speech-understanding tasks [2, 3, 4], and AV information has also been used for training speech-from-video models for applications such as lip reading. [5, 6, 7]

Similar to many related audio, computer-vision, and language-modeling applications, recent AV speech understanding models leverage transformers, and consequently, attention. In standard transformers, the attention mechanisms operate on all input features with a fixed granularity. [8] Context in language, however, is not a regular phenomenon, so it's not obvious that the standard attention approach is optimal in the speech domain. Grouping or pooling features for attention in an adaptive

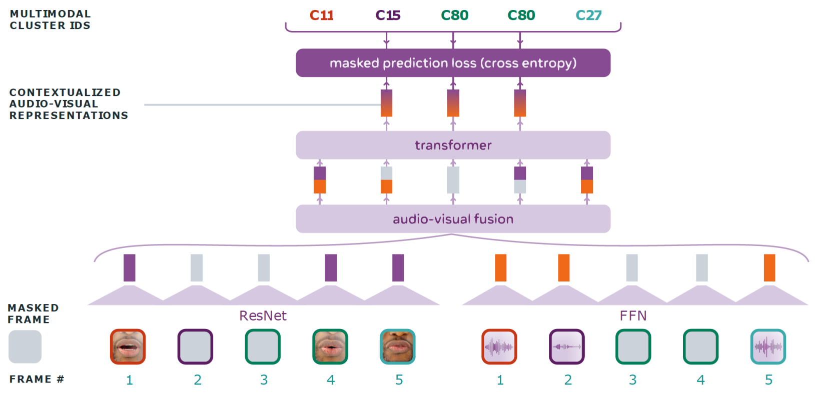


Figure 1: AV-HuBERT model architecture. Figure from [6].

way may provide more flexibility for a model to optimally determine the appropriate context for speech-prediction tasks.

To explore the use of context pooling with attention for AV speech understanding, we build upon the Audio-Visual Hidden-Unit BERT (AV-HuBERT) architecture. [6] As shown in Figure 1, AV-HuBERT's encoder comprises separate audio and video input layers (a linear, feed-forward layer and a modified ResNET-18, respectively) followed by the fusion of audio and visual features, a transformer encoder, and a masked-prediction cross-entropy loss to cluster the input into units of speech. This architecture is based on an audio-only HuBERT model [9, 10] which introduced the use of a BERT-based approach to predict cluster assignments and learn both acoustic and language models from masked, continuous audio inputs. "Modality dropout" is included to optionally zero-out the audio or video features, and allows the model to be fine tuned, for example, for lip reading from video-only input (our focus) or for automatic speech recognition from audio-only input.

Adaptive context pooling (ACP) is a method proposed to address the fixed granularity of standard attention mechanisms. In short, rather than applying the attention query equally to each token in a fixed window of input tokens, the authors in [11] use a two-layer convolutional neural network to adaptively learn a series of attention weights w and pooling sizes s, each of which has n elements where n is the length of the input sequence. As shown in Figure 2, the weights are applied directly to the input sequence, and the pooling sizes are scaled to compute the standard deviations of a family of Gaussian windows g^i which enforce a learned locality on the sequence pooling. The weighted, windowed tokens are then passed to the next attention block. The results in [11] indicate that ACP can be effectively applied to neural machine translation, language modeling, and image classification. Our project was meant to show that ACP also can be successfully applied to AV speech understanding, specifically through integration into the AV-HuBERT decoder for a lip-reading task.

3 Related Work

AV-HuBERT builds on earlier BERT-based approaches for multi-modal masked language modeling including AV-BERT [12], and for (uni-modal) self-supervised speech representation learning including HuBERT [9, 10]. Multi-modal clustering, which is employed in AV-HuBERT for audio-visual feature alignment to units of speech, has been used previously for video action recognition and audio event classification [13].

Prior to AV-HuBERT, the state-of-the-art lip-reading performance was described in [5], in which the authors trained a supervised BLSTM encoder and an LSTM decoder on 31k hours of synchronized audio and video frames extracted from YouTube videos. Other recent AV lipreading models include the conformer-based architecture of Ma *et al.* [14], and the teacher-student approach described by Afouras *et al.* [7] which employs knowledge distillation from an audio-only automatic speech recognition model to a visual speech recognition model.

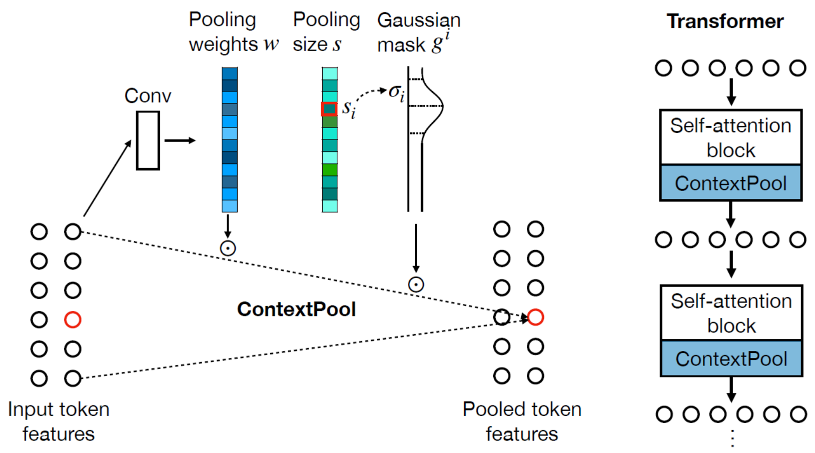


Figure 2: Adaptive context pooling. Adaptively determined pooling weights w are applied directly to input features, and the pooling sizes s are used to define Gaussian masks g^i that enforce a learned locality on the feature pooling. Figure from [11].

The adaptive context pooling we added to AV-HuBERT is meant to enhance standard attention by improving upon the fixed context granularity across features. Other related approaches for feature pooling include area attention [15] in which adjacent features are averaged in groups with learned sizes; local attention [16], in which a learnable Gaussian window is used to select a local set of features; sparse attention [17] in which a sparse subset of features is learned; and multi-scale attention [18] in which a fine-to-coarse attention mechanism with multi-scale spans is used to pool features.

Since the transformer layers in the AV-HuBERT model use standard attention mechanisms, to the best of our knowledge this is the first attempt to apply context pooling to this model.

4 Approach

Our approach for this project involved modifying the attention layers in the AV-HuBERT decoder with ACP to improve on the published results for a lip-reading task. To start, we downloaded pre-trained AV-HuBERT models from [19] and reproduced a subset of results from [6]. Per our milestone report, we achieved word error rates (WERs) of 35.25% with the downloaded "base" AV-HuBERT model (103M parameters) and 27.77% for the "large" AV-HuBERT model (325M parameters), each pre-trained on 1759 hours of unlabeled data from the LRS3 [20] and VoxCeleb2 [21] datasets, and fine-tuned for visual speech recognition on 433 hours of labeled data from LRS3. The corresponding results in the paper are WERs of 34.8% and 28.6%, respectively, suggesting that our training and evaluation infrastructure was properly set up.

To utilize ACP, following [6] we appended a 6-layer multi-head-attention (4 heads) transformer decoder to a pre-trained "base" AV-HuBERT encoder model with 12 transformer blocks for fine-tuning on a lip-reading task.[1] We fine-tuned the encoder-decoder model using the "30-hour" subset of the LRS3 dataset with a train/test/split of 30782/1200/1321 video snippets.

When fine-tuning, AV-HuBERT uses a cross-entropy loss computed on masked subword units,

$$L = -\sum_{t=1}^{s} \log p(w_t | w_{1:t-1}, \mathbf{e}_{1:T}), \tag{1}$$

where $\mathbf{e}_{1:T}$ is the feature (subword) sequence of length T output by the model, $w = w_1, w_2, \ldots, w_s$ is the ground-truth transcription with s sub-word units, and $p(w_t | w_{1:t-1}, \mathbf{e}_{1:T})$ are the target probabilities.

[1]GPU memory restrictions prevented us from working with the large AV-HuBERT encoder model (24 transformer blocks) aside from the initial tests to reproduce published results mentioned earlier.

Our ACP implementation followed the description in [11][2]. The basic idea is depicted in Figure 2: a two-layer convolutional network is applied to a set of language-token features, and used to adaptively generate pooling weights and sizes, the latter of which are used similarly to [16] to generate proximity-enforcing Gaussian pooling windows. The specific details for the convolutional layers are not provided in the paper, so we opted for the first layer to have 1 channel in, 2 channels out, and a kernel size of $(1, e_d/2+1)$, and the second layer to have 2 channels in, 2 channels out, and a kernel size of $(1, e_d/2)$, where e_d is the embedding dimension, which produced the desired output dimensions of 2 channels x sequence length for each sample in the batch. In a subsequent email exchange, the first author of the ACP paper declined to provide further details but indicated that the exact implementation of the convolutional layers was not important. The experiments described below in Section 5 included adding ACP after all 6 transformer layers in the decoder, as well as adding ACP only after the last transformer layer.

5 Experiments

5.1 Data

We used the LRS3 dataset [20] from Oxford University for our experiments.[3] This dataset is a benchmark for AV-HuBERT and other lip-reading models, and contains synchronized audio and video of spoken speech along with annotated transcriptions for determining performance with word error rate (WER). The dataset contains over 9K videos with over 150K utterances, over 4.25M word instances, and a vocabulary size of 70K words, all taken from a database of public TED and TEDx talks that have been annotated.

To utilize LRS3 with AV-Hubert, a number of pre-processing steps were required. These included: splitting long utterances into shorter utterances and generating their time boundaries and labels; trimming the video and audio according to the new time boundaries; extracting the audio from the video files; and generating a list of file IDs and corresponding text transcriptions. A subsequent, necessary feature-extraction process included: detecting facial landmarks in trimmed video files; cropping the videos to the mouth region of interest; counting all the frames per clip; and merging the data back together from the splits that were used for parallel processing.

5.2 Evaluation method

Based on the original analysis of AV-HuBERT in [6] we chose to use WER as out main evaluation metric. The WER is defined as the sum of substitutions, deletions and insertions needed to transform a predicted sentence into the reference sentence, all divided by the total number of words in the reference, and it is a common metric in language prediction models. [22]

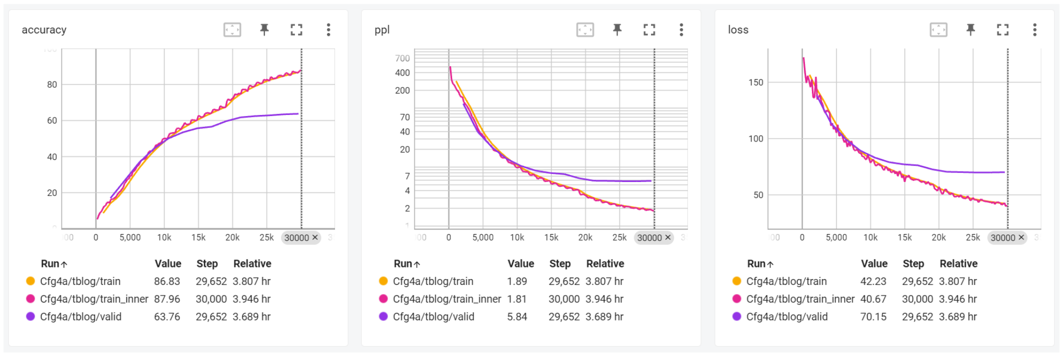
As mentioned in Section 4, the fine-tuning process involved optimizing for masked sub-word unit accuracy with a cross-entropy loss. As discussed below in Section 6, we found that optimizing with one metric but evaluating on another did not lead to intuitive results, for example when both fine-tuning accuracy and inference-time WER were high (*i.e.*, there were not well correlated.) As shown in Figure 3 we also evaluated fine-tuning training performance by monitoring perplexity.

5.3 Experimental details

Our experiments involved modifying the AV-HuBERT decoder, and fine-tuning the entire encoder-decoder model on videos (without audio) for a lip-reading task. The decoder comprises 6 transformer layers, each of which includes multi-head attention acting on the output of the previous layer, as well as multi-head attention acting on the output of the encoder, followed by normalization, dropout, and a fully connected layer. We defined a new transformer layer that included ACP, and evaluated the performance of configurations using ACP in all layers and only in the last layer. The fine-tuning included 30k iterations in most cases and 60k iterations in all other cases, in which the encoder weights were frozen for the first 12k, 19k, 24k or 48k iterations. The initial learning rate (from the

[2]The ACP paper was published without code and we were unable to locate any other existing implementations.

[3]We also intended to use the VoxCeleb2 dataset [21] which was used in the AV-HuBERT paper for pretraining, but shortly after we submitted our project proposal that dataset was removed from its public repository.



Run↑	Value	Step	Relative
🟠 Cfg4a/tblog/train	86.83	29,652	3.807 hr
🔴 Cfg4a/tblog/train_inner	87.96	30,000	3.946 hr
🟣 Cfg4a/tblog/valid	63.76	29,652	3.689 hr

Run↑	Value	Step	Relative
🟠 Cfg4a/tblog/train	1.89	29,652	3.807 hr
🔴 Cfg4a/tblog/train_inner	1.81	30,000	3.946 hr
🟣 Cfg4a/tblog/valid	5.84	29,652	3.689 hr

Run↑	Value	Step	Relative
🟠 Cfg4a/tblog/train	42.23	29,652	3.807 hr
🔴 Cfg4a/tblog/train_inner	40.67	30,000	3.946 hr
🟣 Cfg4a/tblog/valid	70.15	29,652	3.689 hr

Figure 3: Example training/validation curves depicting the accuracy, perplexity (ppl), and loss over time for ACP-enhanced AV-HuBERT.

provided configuration files) was 0.001 but we experienced significant problems with exploding gradients unless we lowered the learning rate or conditioned it through the learning rate scheduler.

In addition to the ACP approach described in [11], we also implemented versions with learned pooling weights but no pooling sizes, and with learned pooling weights plus fixed rather than learned Gaussian windows, mostly for debugging. Specifically, for the fixed Gaussians, for $i \in 0 \cdots (n-1)$ we set σ_i in Figure 2 to $n/10$, where n is the sequence length, rather than computing it from the learned pooling size. We also briefly experimented with alternatives to the 2-layer convolutional network for learning the pooling weights and scores, for example using 1 layer, 4 layers, and 2 layers with the channel expansion after the second layer rather than after the first, but our investigations showed they did not provide substantial improvements and we didn't explore them deeply due to time constraints.

All experiments were run on a workstation with four RTX 3090 GPUs (from which we only used one due to configuration issues with AV-HuBERT code), an AMD Ryzen Threadripper 3990X 64-core processor, and 256GB of RAM.

5.4 Results

The baseline results against which we are comparing are those from Table 1 in [6] with the AV-HuBERT method, Transformer-BASE backbone, S2S criterion, 30 hours of labeled utterances for fine-tuning, and 1759 hours of unlabeled data for pre-training. The reported WER for this case is 46.10%. Our results are shown in Table 2 and the different configurations are summarized in Table 1. "Paper" is from [6], as described above; "Repro" is our reproduction of the published results (no ACP); the remaining configurations were chosen to optimize accuracy, perplexity, and loss whilst avoiding exploding gradients during training.

We see that the best performing models in terms of test-time WER, as indicated by the boldface entries in the table, are the two models that have ACP only in the last layer, using configurations 1 and 4. Configuration 3 and 4d had the best validation performances, however, surprisingly, were not the best at test time. The second best model, as indicated by the underlined entries in the table, was configuration 4b, which included ACP in all layers and only included minor changes from configuration 4a. However, the second best for validation performance were configurations 3 and 4d. We see that the baseline was not the best or the second best for any of the performance metrics except for WER, where it was the second lowest. This is a good sign that ACP was generally, if only moderately, beneficial to model performance. Unfortunately this performance improvement comes at the cost of increasing training time by a factor between 5 and 10.

We expected that by using ACP, performance would increase due to the ability to understand context better for different inputs. We did notice that ACP could perform better than the baseline, however, the performance improvements were not significant and were difficult to obtain through successful trainings. For most cases, performance was worse than expected. We believe this is the case due to some errors and/or misconfigurations in the AV-HuBERT FAIRSEQ [23] open source code, which made the validation metrics uncorrelated to the actual test time performance. This would have had an

Hyperparameter	**Paper**	**Repro**	**Cfg1**	**Cfg2**	**Cfg3**	**Cfg4a**	**Cfg4b**	**Cfg4c**	**Cfg4d**
`fp16`	`true`	`true`	`false`	`false`	`false`	`false`	`false`	`false`	`false`
`max_update`	`30000`	`30000`	`60000`	`60000`	`30000`	`30000`	`30000`	`30000`	`30000`
`update_freq`	`1`	`1`	`2`	`1`	`3`	`3`	`6`	`6`	`6`
`lr`	`0.001`	`0.001`	`8.0e-05`	`5.0e-05`	`0.0002`	`0.00015`	`0.0001`	`0.0001`	`0.0001`
`freeze_finetune_updates`	`24000`	`24000`	`19000`	`48000`	`12000`	`19000`	`19000`	`19000`	`19000`
`warmup_steps`	`10000`	`10000`	`5000`	`10000`	`5000`	`5000`	`5000`	`5000`	`10000`
`final_lr_scale`	`0.05`	`0.05`	`0.2`	`0.2`	`0.2`	`0.2`	`0.05`	`0.2`	`0.2`
`decoder_acp_all_layers`	`false`	`false`	`false`	`false`	`false`	`false`	`true`	`true`	`true`

Table 1: Hyperparameters that were varied in our training configurations.

		Model								
		Baseline		**ACP Last Layer**				**ACP All Layers**		
Metric	**Dataset**	**Paper**	**Repro**	**Cfg1**	**Cfg2**	**Cfg3**	**Cfg4a**	**Cfg4b**	**Cfg4c**	**Cfg4d**
Train time (h) ↓	Train	NR	0.71	6.17	5.62	4.4	**3.7**	7.38	8.76	9.3
Accuracy (%) ↑	Val	NR	63.98	62.42	54.09	68.57	63.76	59.57	63.79	**68.83**
PPL ↓	Val	NR	5.02	6.08	8.44	**4.88**	5.84	6.52	6.15	4.93
Loss ↓	Val	NR	65.78	70.9	79.46	**64.92**	70.15	73.19	71.57	65.77
WPS ↑	Test	NR	123	119	120	126	**133**	106	117	107
Tokens/s ↑	Test	NR	127.29	123.13	123.68	130.22	**137.39**	108.54	119.59	109.66
WER (%) ↓	Test	46.10	46.10	**45.07**	51.85	47.98	**45.07**	47.23	56.35	55.60

Table 2: Lip-reading results for our ACP-enhanced AV-HuBERT model. See Table 1 for an explanation of the different model configurations. NR = not reported. Boldface = best. Underlined = second best.

affect on the best checkpoint saving criterion. Additionally, this tells us that the approach for ACP needs to be built on stable foundations but, however, could have promising potential for AV-HuBERT.

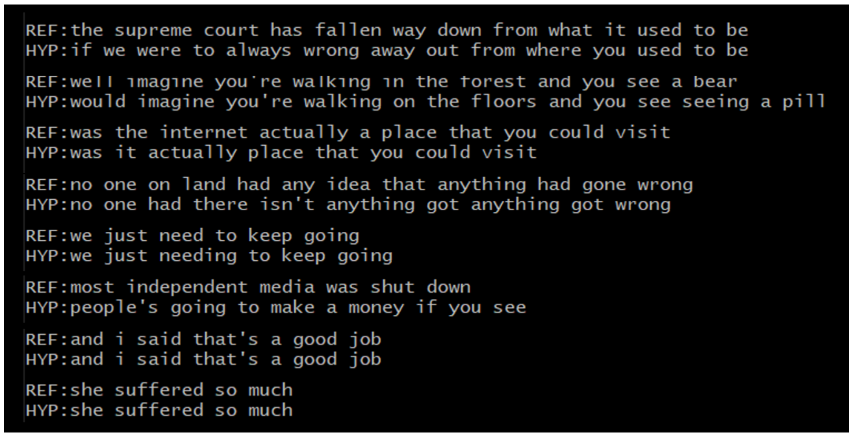
6 Analysis

During the fine-tuning stage, accuracy is computed as the number of *correct* masked unigram-based subword units [24] divided by the *total* number of masked unigram-based subword units accumulated over the training sequences (sentences). During inference, model performance is reported in terms of the word error rate (WER) of an output from sequence generation. While intuitively these two metrics should be well correlated, we found this often not to be the case. For example, Configuration 4a in Table 2 gave us the best performance at inference time, but was not competitive in training. Conversely, Configuration 4d had the highest training accuracy but also a high WER. It has been suggested elsewhere that training through minimizing WER, or a reasonable proxy of it, might be more effective [25] but we did not have a chance to explore that.

We found the exploding gradients were a difficult problem to overcome. We employed ReLU activation functions, used batch normalization and performed gradient clipping, none of which had a noticeable effect on the exploding gradients. The only two mitigations we found to be effective were increasing the bit depth to 32-bit from 16-bit, and lowering the learning rate or modifying the learning rate scheduler to avoid an explosion of gradients. These mitigations, however, slowed down training time and also made it more difficult to search for optimal configurations for the models and training.

In regard to ACP, we also tried implementations with non-adaptive Gaussian windows and noticed that training metrics performed significantly well, however, these models did not generalize to the test set. From this, we discovered that errors in the public FAIRSEQ and AV-HuBERT implementations prevented correct validation/inference of the models during training in certain cases.

After avoiding the above issues, we can see from Table 2 that ACP performs best when it is only in the last layer of the network as opposed to all layers. Overall, we find that ACP improves performance of the model above that reported in the SOTA paper for the 'BASE' model. Example outputs from our best model can be seen in Figure 4. The model shows good predictions that match the ground truth, which is impressive considering the only input modality is video alone. The output could likely be improved with a better language model to improve sentence structure and reduce the likelihood of incorrect words that are themselves unlikely in the sentence and context. Tweaking the



```
REF:the supreme court has fallen way down from what it used to be
HYP:if we were to always wrong away out from where you used to be

REF:well imagine you're walking in the forest and you see a bear
HYP:would imagine you're walking on the floors and you see seeing a pill

REF:was the internet actually a place that you could visit
HYP:was it actually place that you could visit

REF:no one on land had any idea that anything had gone wrong
HYP:no one had there isn't anything got anything got wrong

REF:we just need to keep going
HYP:we just needing to keep going

REF:most independent media was shut down
HYP:people's going to make a money if you see

REF:and i said that's a good job
HYP:and i said that's a good job

REF:she suffered so much
HYP:she suffered so much
```

Figure 4: Example inference output from our best performing model. REF = reference. HYP = hypothesis (prediction).

hyperparameters of ACP could also help, where adjustments to the scale of the standard deviation of the Gaussian weights, or even the statistical curves, could be better matched to the dataset or context task.

We can see from the examples in Figure 4 that for several sentences the model gets a perfect match, e.g. “and i said that’s a good job” and “she suffered so much”. On the other hand, there are instances where contextual understanding could be improved to help avoid misclassifications of words, *e.g.*, “we just *needing* to keep going” could be improved by understanding that “needing” does not fit in this context. We can see that the model still outputs sensible sentences with words that fit context together even if they are wrong, *e.g.*, “was it actually place that you could visit” misunderstands the words “the internet actually” and uses “it actually”, which still makes sense in the context of the sentence. This also occurs in the sentence “would imagine you’re walking on the floors and you see seeing a pill”, where if you were on the floor you would more likely see a pill than a bear, and “floors” was misunderstood from “forest”, which look similar for an input of video only.

7 Conclusion

For this project we chose to add adaptive context pooling to the transformers in the decoder of the AV-HuBERT architecture with the intention of improving the model’s performance, in terms of word error rate, on a lip-reading task (speech from video of a talker). We faced a few challenges including a time-intensive data pre-processing routine, a code base that was more complex than we expected, lack of detail in the context-pooling paper, and an unstable implementation that required significant effort to prevent exploding gradients. Our best model very slightly improved on the baseline: we obtained a WER of 45.07% while the baseline achieved 46.10%. This small improvement came at the cost of significantly increased training time, although inference was slightly faster (based on a single pass through the test data.)

Several factors may have contributed to the lack of improvement over the baseline, and we have considered three avenues for future work. First, despite the comments from the first author of [11] it’s unclear how important the architecture of the convolutional layers in the ACP approach is, particularly in this context, so we would like to implement and evaluate more options for adaptively computing the pooling weights and sizes. Second, it would be valuable to explore applying ACP to each head of the multi-head attention in the AV-HuBERT decoder. This would have required modifying and rebuilding the pre-compiled C++ attention library from PyTorch that is used by AV-HuBERT, but this was beyond the scope of this project. Finally, there are some idiosyncrasies in the FAIRSEQ sequence-modeling toolkit upon which AV-HuBERT is built that made debugging difficult, so gaining a deeper understanding of this could help improve our ACP-enhanced AV-HuBERT implementation and its lip-reading performance.

8 Contributions

We conceived the project idea together, jointly wrote the proposal, milestone report, and final report, and jointly prepared the poster. We both independently pre-processed the LRS3 dataset and ran pre-trained versions of the AV-HuBERT model to create a baseline and test our infrastructure. Paul implemented the initial version of adaptive context pooling (ACP) and Jacob optimized it. Jacob developed the infrastructure to fine-tune the ACP-enhanced AV-HuBERT model and ran our test cases. Both of us investigated a problem with exploding gradients that delayed our final model runs.

References

[1] Bowen Shi, Wei-Ning Hsu, and Abdelrahman Mohamed. Robust self-supervised audio-visual speech recognition. *arXiv preprint arXiv:2201.01763*, 2022.

[2] Qiya Song, Bin Sun, and Shutao Li. Multimodal sparse transformer network for audio-visual speech recognition. *IEEE Transactions on Neural Networks and Learning Systems*, 2022.

[3] Triantafyllos Afouras, Joon Son Chung, Andrew Senior, Oriol Vinyals, and Andrew Zisserman. Deep audio-visual speech recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(12):8717–8727, 2018.

[4] Bo Xu, Cheng Lu, Yandong Guo, and Jacob Wang. Discriminative multi-modality speech recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14433–14442, 2020.

[5] Takaki Makino, Hank Liao, Yannis Assael, Brendan Shillingford, Basilio Garcia, Otavio Braga, and Olivier Siohan. Recurrent neural network transducer for audio-visual speech recognition. In *2019 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*, pages 905–912, 2019.

[6] Bowen Shi, Wei-Ning Hsu, Kushal Lakhotia, and Abdelrahman Mohamed. Learning audio-visual speech representation by masked multimodal cluster prediction. *arXiv preprint arXiv:2201.02184*, 2022.

[7] Triantafyllos Afouras, Joon Son Chung, and Andrew Zisserman. ASR is all you need: Cross-modal distillation for lip reading. In *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 2143–2147, 2020.

[8] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in Neural Information Processing Systems*, 30, 2017.

[9] Wei-Ning Hsu, Benjamin Bolte, Yao-Hung Hubert Tsai, Kushal Lakhotia, Ruslan Salakhutdinov, and Abdelrahman Mohamed. HuBERT: Self-supervised speech representation learning by masked prediction of hidden units. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 29:3451–3460, 2021.

[10] Wei-Ning Hsu, Yao-Hung Hubert Tsai, Benjamin Bolte, Ruslan Salakhutdinov, and Abdelrahman Mohamed. HuBERT: How much can a bad teacher benefit ASR pre-training? In *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 6533–6537, 2021.

[11] Chen Huang, Walter Talbott, Navdeep Jaitly, and Joshua M Susskind. Efficient representation learning via adaptive context pooling. In *International Conference on Machine Learning*, pages 9346–9355. PMLR, 2022.

[12] David M Chan, Shalini Ghosh, Debmalya Chakrabarty, and Björn Hoffmeister. Multi-modal pre-training for automated speech recognition. In *ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 246–250, 2022.

[13] Humam Alwassel, Dhruv Mahajan, Bruno Korbar, Lorenzo Torresani, Bernard Ghanem, and Du Tran. Self-supervised learning by cross-modal audio-video clustering. *Advances in Neural Information Processing Systems*, 33:9758–9770, 2020.

[14] Pingchuan Ma, Stavros Petridis, and Maja Pantic. End-to-end audio-visual speech recognition with conformers. In *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 7613–7617. IEEE, 2021.

[15] Yang Li, Lukasz Kaiser, Samy Bengio, and Si Si. Area attention. In *International Conference on Machine Learning*, pages 3846–3855. PMLR, 2019.

[16] Baosong Yang, Zhaopeng Tu, Derek F Wong, Fandong Meng, Lidia S Chao, and Tong Zhang. Modeling localness for self-attention networks. *arXiv preprint arXiv:1810.10182*, 2018.

[17] Aurko Roy, Mohammad Saffar, Ashish Vaswani, and David Grangier. Efficient content-based sparse attention with routing transformers. *Transactions of the Association for Computational Linguistics*, 9:53–68, 2021.

[18] Zihao Ye, Qipeng Guo, Quan Gan, Xipeng Qiu, and Zheng Zhang. BP-Rransformer: Modelling long-range context via binary partitioning. *arXiv preprint arXiv:1911.04070*, 2019.

[19] Pretrained AV-HuBERT Models. `https://facebookresearch.github.io/av_hubert/`, 2022. [Online; accessed 14-Feb-2023].

[20] Triantafyllos Afouras, Joon Son Chung, and Andrew Zisserman. LRS3-TED: A large-scale dataset for visual speech recognition. *arXiv preprint arXiv:1809.00496*, 2018.

[21] J. S. Chung, A. Nagrani, and A. Zisserman. VoxCeleb2: Deep speaker recognition. In *INTERSPEECH*, 2018.

[22] Ahmed Ali and Steve Renals. Word error rate estimation for speech recognition: e-WER. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 20–24, 2018.

[23] Myle Ott, Sergey Edunov, Alexei Baevski, Angela Fan, Sam Gross, Nathan Ng, David Grangier, and Michael Auli. FAIRSEQ: A fast, extensible toolkit for sequence modeling. In *Proceedings of NAACL-HLT 2019: Demonstrations*, 2019.

[24] Taku Kudo. Subword regularization: Improving neural network translation models with multiple subword candidates. *arXiv preprint arXiv:1804.10959*, 2018.

[25] Rohit Prabhavalkar, Tara N. Sainath, Yonghui Wu, Patrick Nguyen, Zhifeng Chen, Chung-Cheng Chiu, and Anjuli Kannan. Minimum word error rate training for attention-based sequence-to-sequence models. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2018.