Effective Context in Neural Speech Models

Yen Meng, Sharon Goldwater, Hao Tang

The Centre for Speech Technology Research, University of Edinburgh, United Kingdom

yen.meng@ed.ac.uk, sgwater@inf.ed.ac.uk, hao.tang@ed.ac.uk

Abstract

Modern neural speech models benefit from having longer context, and many approaches have been proposed to increase the maximum context a model can use. However, few have attempted to measure how much context these models actually use, i.e., the effective context. Here, we propose two approaches to measuring the effective context, and use them to analyze different speech Transformers. For supervised models, we find that the effective context correlates well with the nature of the task, with fundamental frequency tracking, phone classification, and word classification requiring increasing amounts of effective context. For self-supervised models, we find that effective context increases mainly in the early layers, and remains relatively short—similar to the supervised phone model. Given that these models do not use a long context during prediction, we show that HuBERT can be run in streaming mode without modification to the architecture and without further fine-tuning.

Index Terms: interpretability, self-supervised learning, probing

1. Introduction

The recent success of speech and language processing systems can be largely attributed to better modeling of context. Various speech tasks benefit from contextualized speech embeddings learned with self-supervision [1–5]. Automatic speech recognition, summarization, question answering, and language modeling have all been shown to improve when models are engineered to have a wider context [6–11]. However, observing an improved performance does not necessarily imply that a wider context is actually used. Failing to properly attribute where the improvement is from gives the false impression that longer context is always better. It is possible that a wider context is not even necessary to achieve competitive performance.

In this work, we distinguish between the context a model has access to (i.e., the context by design) and the context that a model actually uses (the effective context). The amount of context available to the model can be increased by changing the model architecture [9–11]. However, the context that the model actually uses will be affected by various other design choices, such as the data and the training algorithm. LSTMs, for example, have access to the entire history but also have difficulty making full use of it [12]. Several studies have presented indirect evidence that Transformers, despite having access to the entire input sequence, do not make use of the full context. They approach this problem by modifying the model architecture [13–15] or by analyzing attention maps [13, 16]. A few other studies [17–21], though not directly studying the amount of context, have probing results that suggest the limited use of context in Transformers. Overall, these approaches are ad-hoc and limited either to a particular model architecture or to a particular task.

In this work, we approach the problem of measuring effective context from first principles. If a change to a part of the input does not affect the output much, then that part of the input is effectively not used by the model and hence not in the effective context. We propose two approaches, one based on truncation of the input and the other based on the Jacobian with respect to the input. The truncation approach is more intuitive—truncating input frames not in the effective context should hardly affect the output. The Jacobian approach (similar to gradients) measures the effect of infinitesimal changes on the input. Both approaches are model-agnostic, not tied to specific loss functions, do not require labels, and can be applied to any layer of a network.

We apply the proposed approaches to measure effective context of speech Transformers, as they are the dominant architecture in both supervised and self-supervised settings. In the supervised setting, we find that the amount of effect context used in Transformers for predicting f0, phones, and words increases in that order. In the self-supervised setting, we find that the effective context of self-supervised Transformers, specifically wav2vec 2.0 [3], HuBERT [4], and WavLM [5], is shorter than that for predicting phones in the supervised setting. We also observe a clear correlation between the amount of effective context and probing performance on phones and words.

Given the relatively short effective context of self-supervised Transformers, limiting the history and lookahead of these models should not hurt their performance much. This implies that we can readily run pretrained Transformers in low-latency (400ms) streaming mode without any modification to the architecture or further training. Using a simple probing classifier, our experiments show that, compared to a phone error rate of 11.9% when using full utterances, we only observe a 0.6% degradation with the full history, and a 1.5% degradation with a 2s history.

2. Measuring Effective Context

We define effective context based on the following principle. For an input utterance of T frames $x_1, \ldots, x_T$, we are interested in how a function f changes as we change the frames. At a high level, if changes made to frame x_t do not change the output of f by much, then x_t is not part of the effective context of f. The function f can be the computation from the input to any of the intermediate layers, or all the way to the loss. The output of f typically has multiple frames, i.e., $h_1, \ldots, h_T = f(x_1, \ldots, x_T)$, and we will use $[f(x_1, \ldots, x_T)]_t$ to denote h_t. Below we detail two changes that can made to the frames, leading to the truncation approach and the Jacobian approach.

2.1. The Truncation Approach

If we are interested in whether a frame x_t is in the effective context or not, the most intuitive approach is to remove x_t from

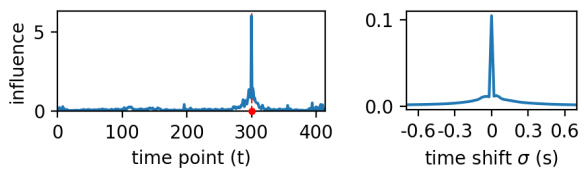

Figure 1: *Examples illustrating influence. Left: The x-axis is the time point τ of the input utterance x, and the y-axis is the calculated* ***influence*** *value $s(t, \tau)$ for the timepoint $t = 300$ (red dot). Right: The* ***relative influence*** *$S(\sigma)$ (normalized), where the x-axis is the time shift to the center frame.*

.

the input to f. Removing single frames is not very meaningful as, in speech, a lot can be inferred from neighboring frames. Instead, for models that can make use of the full context, we make the assumption (which we will later verify) that f has a symmetric effective context. We choose a window of size $2W + 1$ frames centered at t and truncate the frames outside of the window. We then compute

$$d\left(\Big[f(x_1, \ldots, x_T)\Big]_t, \Big[f(x_{t-W}, \ldots, x_{t+W})\Big]_t\right) \quad (1)$$

for $t = 1, \ldots, T$ of an utterance and average over a data set, where d is a distance function that can be the ℓ_2 distance or a performance metric, such as phone probing error rate. The truncation approach has been inspired by similar ideas used to study the context of text-based language models [22–24].

2.2. The Jacobian Approach

Instead of explicitly truncating, another approach is to make changes to the input. Formally, we want to compute

$$\left\| \Big[f(\ldots, x_\tau, \ldots)\Big]_t - \Big[f(\ldots, x_\tau + \epsilon, \ldots)\Big]_t \right\|_{\mathrm{F}} \quad (2)$$

where ϵ is the change made to the frame $x_\tau \in \mathbb{R}^K$. When ϵ is infinitesimal, the above term becomes the Frobenius norm of the Jacobian (or the ℓ_2 norm of the gradient when f is $\mathbb{R}^K \to \mathbb{R}$). We call this quantity, denoted as $s(t, \tau)$, the **influence** of x_τ on h_t (or $[f(x_1, \ldots, x_T)]_t$). See Fig. 1 (left) for an example. Note that, compared to the truncation approach, t and τ are decoupled here, and no windowing or symmetry assumptions are made.

In practice, most automatic differentiation packages do not explicitly calculate the full Jacobian matrix but rather the vector-Jacobian product. To compute the full Jacobian matrix of a hidden vector $h_t \in \mathbb{R}^D$, we simply compute the vector-Jacobian product on the D-dimensional standard basis, backpropagating D times. In other words, the influence is computed as

$$s(t, \tau) = \sqrt{\sum_{k=1}^{K} \sum_{d=1}^{D} \left(e_d^\top \frac{\partial h_t}{\partial x_{\tau,k}} \right)^2} \quad (3)$$

where e_d is d-th element of the D-dimensional standard basis, and $\frac{\partial h_t}{\partial x_\tau}$ is the Jacobian matrix of time t with respect to the input at time τ.

Similar to the truncation approach, we introduce windowing and study the influence relative to the center frame. For a data set of N utterances, each of which has T_n frames with the measured influence $s_n(t, \tau)$ of f, for $n = 1, \ldots, N$, the **relative influence** at time shift σ is then defined as

$$S(\sigma) = \sum_{n=1}^{N} \sum_{t=1}^{T_n} s_n(t, t + \sigma). \quad (4)$$

For example, when $S(-w)$ is high, the influence on hidden vectors from the input that is w frames before is, on average, high. Given a window size of $2W + 1$ frames, we compute the relative influence for time shifts $\sigma = -W, -W + 1, \ldots, W$. The Jacobian matrices are not scale invariant, so we normalize the relative influence such that $\sum_{\sigma=-W}^{W} S(\sigma) = 1$. See Fig. 1 (right) for an example. After normalization, the relative influence of different models of layers can be compared. In the relative influence plots, we convert frames into seconds for easier reference.

To contrast with other approaches, saliency [25] of certain parts of the input used to explain model prediction is typically computed by taking the gradient to the input. Though the mathematical formulation (Eq. 3) of the Jacobian approach happens to be a generalization of saliency, our goal is different—to estimate the effective context rather than explain particular outputs—and the generalization allows us to apply the approach to any layer (not just the final output). Other explainability methods, such as LIME [26] and SHAP [27], are not suitable for our purpose, as they do not necessarily inform us about the effective context, are not task-agnostic, or cannot be applied to intermediate layers.

3. Experiments

Since there isn't any prior work on measuring effective context in speech models, our goal is to provide a foundation for future work to measure the effective context for a broader range of models and architectures. In this section, we select a few models to analyze and showcase the utility of our approach.

3.1. Pilot experiments with truncation

For the first set of experiments, we measure the effective context of a pretrained HuBERT (base)[1]. We follow [1, 2, 28] and train a linear probing classifier (using multinomial logistic regression) on WSJ to predict phones for each layer of HuBERT. We then measure how the phone error rate changes as we truncate the frames and only keep a window of frames. This measures how much change in the hidden vectors is needed to flip the label of a classifier. We also directly measure the ℓ_2 distance between the representations computed from the original and truncated inputs.

Results on `dev93` are shown in Fig. 3. Although very narrow truncation windows have a large effect, by the time the window size reaches 2.0s, PER is similar for truncated and un-truncated input across all layers. The same is true for ℓ_2 with a window size of 4.0s. We conclude that the effective context of HuBERT is about 2.0s on either side of the center frame (and as a reference the average utterance length in `dev93` is 7.8s).

Based on the trend in ℓ_2 distance, it seems that the effective context increases after each layer, though the trend is less clear in the PER case. It is relatively difficult to summarize the layerwise results with the truncation approach, but these initial results will be useful later as we check for consistency.

[1]The average utterance length of LibriSpeech where HuBERT is trained on is 12.3s. Each audio example is cropped to 15.6s during pretraining.

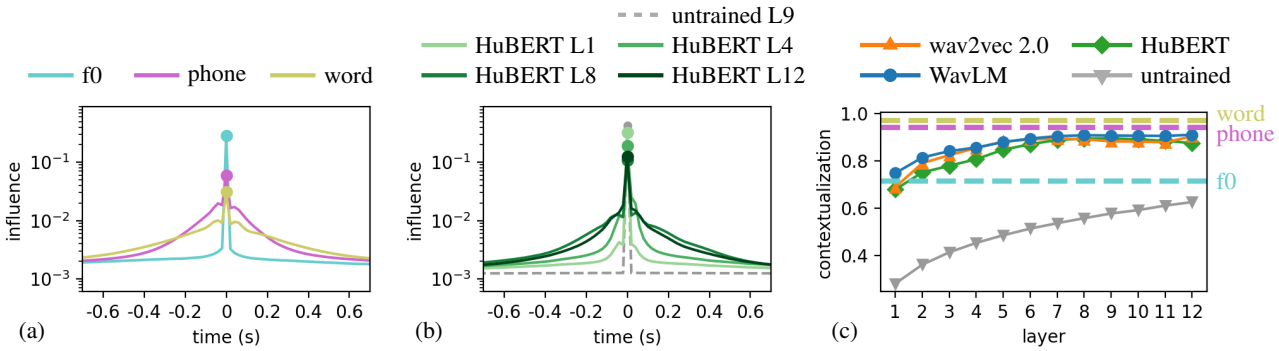

Figure 2: *(a) Relative influence in the final layers of supervised 6-layer Transformer models trained for different tasks. The y-axis is on a log scale and the x-axis is only shown between ±0.7s, although the relative influence values were computed with a window size of 5 seconds on both sides. Dots show the heights of the center peaks. (b) As in (a) but for different layers of HuBERT. (c) Contextualization of different models and layers. The horizontal lines are values of the supervised models on the final layer.*

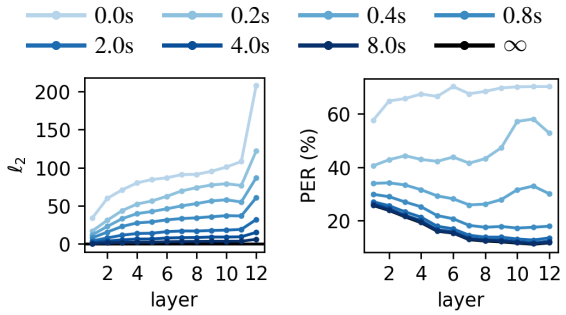

Figure 3: *The change of output in terms of the ℓ_2 distance (left) and the phone error rates (right), as we vary the window size of input to HuBERT (different coloured lines).*

3.2. Results with the Jacobian approach

We follow the same setting as the truncation approach. However, instead of immediately comparing results on HuBERT layers, we first analyze a set of supervised Transformers for predicting f0, phones, and words. We will then use these results to contextualize the Jacobian results on HuBERT layers, and further extend them to wav2vec 2.0 and WavLM. We compute the relative influence with a 10s window, sufficiently large given that the ℓ_2 does not change much already within a 4.0s window.

Effective context of supervised models For all supervised models, we use the same 6-layer Transformer with sinusoidal position encoding. Training is done on the `si284` subset of WSJ. The input to the Transformers is 80 dimensional, consisting of 40-dimensional Mel spectrograms stacked every two frames. The frame rate is 20ms, matching those of HuBERT, wav2vec 2.0, and WavLM. For f_0 tracking, the targets are extracted with the PYIN algorithm [29]. Phone and word classification are conducted at the frame level, following [12], with labels obtained through forced alignments computed with an HMM-GMM from Kaldi. For reference, these models achieve 8.2 Hz RMSE on f_0 tracking, a 12.4% error rate on phone classification, and a 26.2% error rate on word classification on `dev93`.

In Fig. 2(a), we show the relative influence of the final layer of the 6-layer Transformer, as we are interested in the effective context needed for the tasks. First, the relative influence in all three tasks is roughly symmetric and drops quickly as we move away from the center. This suggests that most of the influence comes from the center frame (and showing the range within ±0.7s in Fig. 2 is already sufficient). Second, predicting f0 uses the least amount of effective context, followed by predicting phones. Predicting words uses the largest effective context, as can be seen from the lower center peak and the fatter tail.

Effective context of self-supervised models For the self-supervised setting, we focus on masked prediction models, HuBERT, wav2vec2.0, and WavLM, which are 12-layer Transformers pretrained on the 960-hour subset of LibriSpeech (base models). These models take wave samples as input, so we measure the norms at the output of the convolution feature extractor (i.e., the input before the Transformer encoder) so that the context measurement for all models only includes Transformer layers.

Fig. 2(b) shows the relative influence of layer 1, 4, 8, 12, compared to a late layer in an untrained Transformer. We again see that the relative influence for all layers decays sharply moving away from the center and the shape is roughly symmetric. The effective context seems to increase through the layers of the pretrained model, but remains narrow for the untrained model.

We then compare the HuBERT layers in Fig. 2(b) with the supervised results in Fig. 2(a). The effective context of higher layers of HuBERT is close to the 6-layer phone model but much smaller compared to the word model. This might explain a recent finding that self-supervised speech representations are more phonetic than semantic [21]. The results presented above were on WSJ `dev93`, but we also analyzed relative influence for HuBERT on LibriSpeech `test-clean`(average utterance length of 7.4s), with very similar results.

Layerwise analysis Though relative influence provides a detailed shape of the effective context, it is difficult to compare quantitatively among layers and across models. For this, it is desirable to summarize the relative influence curve using a single value. While there are various options (such as reporting the width that contains some threshold proportion of the mass), the values depend both on the choice of threshold and on the window size used to compute the relative influence.

In practice, we find that a very simple statistic works well: we use $1 - S(0)$, where $S(0)$ is the relative influence at the center frame, and refer to this as the **contextualization**. It is 0 when only the immediate input frame influences the result and approaches 1 if there is almost no influence from the center frame. The exact values still depend on the window size used to compute the relative influence, but we can compare relative values of contextualization for different models/layers, as long

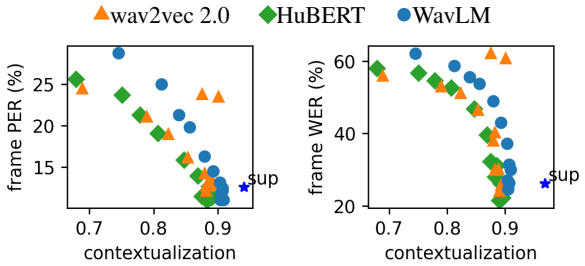

Figure 4: *Relation between contextualization and probing performance on phones (left) and words (right). Each point represents a layer of a model. The last layer of the supervised 6-layer Transformer is annotated.*

as all values are based on the same window size (as here).[2]

Fig. 2(c) shows the contextualization of wav2vec 2.0, HuBERT, WavLM, and an untrained Transformer (randomly initialized following [30]). The contextualization of pretrained models increases from layer 1 to layer 7, then plateaus. When we include the contextualization of the 6-layer phone and word predictor, it is now clear that the pretrained models are less contextualized than the phone predictor. Interestingly, the effective context of the untrained model increases throughout the layers. This potentially explains the increase in performance throughout the layers for speaker tasks in untrained Transformers [31]. Comparing the pretrained models and the untrained one, it is clear that self-supervised training is a process of learning contextualization.

Fig. 4 illustrates the relation between phone and word probing (again using multinomial logistic regression classifiers) and contextualization. We observe a clear correlation between lower error rates and greater contextualization. The only exceptions to this general pattern are the last two layers of the wav2vec 2.0 model (the two outlier points in the plots), which other studies have also found to behave idiosyncratically [17, 18, 31]. Fig. 4 suggests that higher contextualization does not guarantee higher probing accuracy; rather, there seems to be a minimum amount of effective context that a model needs to achieve in order to predict phones and words well. The phone error rate decreases steadily starting from the first layer, whereas for predicting words, the probing error rate remains high in the first few layers, and starts to drop once the model has reached a certain degree of contextualization.

4. Simulating a Streaming HuBERT

Given that the effective context of pretrained Transformers is not long, we should be able to truncate their context and run them in a low-latency streaming mode without much performance loss. This would give us a streaming representation extractor. In contrast to other streaming models [32–35], this approach does not require modifications to the architecture or attention and does not require further training. We take HuBERT as an example and run it as a sliding window over the input frames. For streaming, the choice to make is the amount of history and the amount of lookahead to use. The history largely determines the overall memory usage, while the lookahead determines the latency. The window of input to HuBERT is the concatenation of the history and the lookahead.

Fig. 5 shows the results of phone probing on layer 9 of a streaming HuBERT, using the probing classifier from §3.1 without retraining. For an unlimited history (right panel of Fig. 5), a lookahead of 400ms is sufficient to recover most of the phone error rates, achieving 12.5% compared to 11.9% when using the full context. In terms of the history to truncate if we fix the lookahead to 400ms (left panel of Fig. 5, lowest curve), a history of 2s is sufficiently strong, achieving 13.7%. The results suggest that HuBERT representations can be extracted using history and lookahead values within the range used for recent streaming ASR systems [32–35], with little loss of fidelity.

[2]We found that qualitative trends for contextualization depended less on window size than other statistics we considered early on.

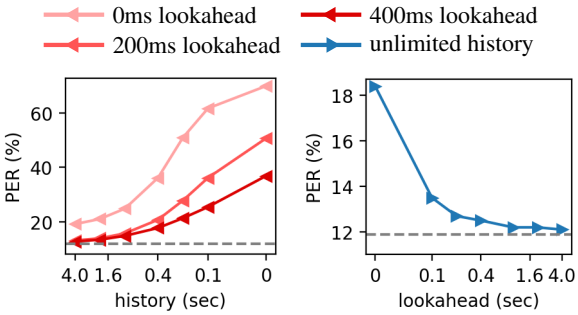

Figure 5: *Results of different streaming settings. The dashed line represents the probing error rate with full context. We vary the number of lookahead with unlimited history (right), and vary the number of history with different amounts of lookahead (left).*

5. Discussion and Conclusion

In this work, we designed two complementary approaches to measuring effective context from first principles. The truncation approach is a direct modification to the input and can be easily verified by task performance difference, but how and how much we can truncate requires empirical verification. The Jacobian approach is less clear in terms of task performance as changes to the input are not actually made. However, it does not require any assumptions regarding the shape of the relative influence curve, and allows us to examine these directly, providing a more fine-grained view of the influence of context at different distances from the current frame. Both approaches are easy to implement, trivially parallelizable, and can be applied regardless of architecture or training objective.

By combining these two approaches, we found strong evidence that effective context differs for different supervised tasks using the same model architecture, and that self-supervised Transformers have relatively short effective context. Our results also indicate that for pretrained models, there is a strong correlation between contextualization and probing performance. Nevertheless, contextualization alone cannot predict performance. In particular, our 6-layer supervised models have a larger effective context than the 12-layer pretrained models, but have slightly worse phone and word error rates. This implies that other factors, such as the structure of the representations or the amount of training data, are also important. Our work, however, already provides us with sufficient insights into how self-supervised models use context and to have practical applications. We are able to simulate a streaming HuBERT with low latency and minor performance degradation. In future work, we hope to use the tools developed here to further examine effective context in other types of models, such as ECAPA-TDNN [36] or Whisper [37]. Designing models that have large context windows does not necessarily contribute to solving the long-context modeling problem unless models actually use that context, and we hope that this work offers an opportunity to design long-context models from first principles.

6. References

[1] Y.-A. Chung, W.-N. Hsu, H. Tang, and J. Glass, "An unsupervised autoregressive model for speech representation learning," in *INTERSPEECH*, 2019.

[2] A. v. d. Oord, Y. Li, and O. Vinyals, "Representation learning with contrastive predictive coding," *arXiv preprint arXiv:1807.03748*, 2018.

[3] A. Baevski, Y. Zhou, A. Mohamed, and M. Auli, "wav2vec 2.0: A framework for self-supervised learning of speech representations," *Advances in NeurIPS*, 2020.

[4] W.-N. Hsu, B. Bolte, Y.-H. H. Tsai, K. Lakhotia, R. Salakhutdinov, and A. Mohamed, "Hubert: Self-supervised speech representation learning by masked prediction of hidden units," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 2021.

[5] S. Chen, C. Wang, Z. Chen, Y. Wu, S. Liu, Z. Chen, J. Li, N. Kanda, T. Yoshioka, X. Xiao *et al.*, "Wavlm: Large-scale self-supervised pre-training for full stack speech processing," *IEEE Journal of Selected Topics in Signal Processing*, 2022.

[6] T. Hori, N. Moritz, C. Hori, and J. L. Roux, "Transformer-Based Long-Context End-to-End Speech Recognition," in *INTERSPEECH*, 2020.

[7] R. Masumura, T. Tanaka, T. Moriya, Y. Shinohara, T. Oba, and Y. Aono, "Large context end-to-end automatic speech recognition via extension of hierarchical recurrent encoder-decoder models," in *ICASSP*, 2019.

[8] R. Flynn and A. Ragni, "How much context does my attention-based asr system need?" in *INTERSPEECH*, 2024.

[9] Z. Dai, Z. Yang, Y. Yang, J. Carbonell, Q. V. Le, and R. Salakhutdinov, "Transformer-xl: Attentive language models beyond a fixed-length context," in *ACL*, 2019.

[10] T. Munkhdalai, M. Faruqui, and S. Gopal, "Leave no context behind: Efficient infinite context transformers with infini-attention," *arXiv preprint arXiv:2404.07143*, 2024.

[11] A. Mohtashami and M. Jaggi, "Landmark attention: Random-access infinite context length for transformers," *Advances in NeurIPS*, 2023.

[12] H. Tang and J. Glass, "On training recurrent networks with truncated backpropagation through time in speech recognition," in *SLT*, 2018.

[13] S. Zhang, E. Loweimi, P. Bell, and S. Renals, "On the usefulness of self-attention for automatic speech recognition with transformers," in *SLT*, 2021.

[14] Y. Peng, S. Dalmia, I. Lane, and S. Watanabe, "Branchformer: Parallel mlp-attention architectures to capture local and global context for speech recognition and understanding," in *ICML*, 2022.

[15] T. Parcollet, R. van Dalen, S. Zhang, and S. Bhattacharya, "Sumformer: A linear-complexity alternative to self-attention for speech recognition," in *INTERSPEECH*, 2024.

[16] K. Shim, J. Choi, and W. Sung, "Understanding the role of self attention for efficient speech recognition," in *ICLR*, 2022.

[17] A. Pasad, J.-C. Chou, and K. Livescu, "Layer-wise analysis of a self-supervised speech representation model," in *ASRU*, 2021.

[18] A. Pasad, B. Shi, and K. Livescu, "Comparative layer-wise analysis of self-supervised speech models," in *ICASSP*, 2023.

[19] A. Pasad, C.-M. Chien, S. Settle, and K. Livescu, "What do self-supervised speech models know about words?" *TACL*, 2023.

[20] M. de Seyssel, M. Lavechin, Y. Adi, E. Dupoux, and G. Wisniewski, "Probing phoneme, language and speaker information in unsupervised speech representations," in *INTERSPEECH*, 2022.

[21] K. Choi, A. Pasad, T. Nakamura, S. Fukayama, K. Livescu, and S. Watanabe, "Self-supervised speech representations are more phonetic than semantic," in *INTERSPEECH*, 2024.

[22] U. Khandelwal, H. He, P. Qi, and D. Jurafsky, "Sharp nearby, fuzzy far away: How neural language models use context," in *ACL*, 2018.

[23] J. O'Connor and J. Andreas, "What context features can transformer language models use?" in *ACL*, 2021.

[24] M. Levy, A. Jacoby, and Y. Goldberg, "Same task, more tokens: the impact of input length on the reasoning performance of large language models," *ACL*, 2024.

[25] K. Simonyan, A. Vedaldi, and A. Zisserman, "Deep inside convolutional networks: Visualising image classification models and saliency maps," in *ICLR*, 2014.

[26] M. T. Ribeiro, S. Singh, and C. Guestrin, "" why should i trust you?" explaining the predictions of any classifier," in *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, 2016, pp. 1135–1144.

[27] S. Lundberg, "A unified approach to interpreting model predictions," *NeurIPS*, 2017.

[28] G.-P. Yang, S.-L. Yeh, Y.-A. Chung, J. Glass, and H. Tang, "Autoregressive predictive coding: A comprehensive study," *IEEE Journal of Selected Topics in Signal Processing*, 2022.

[29] M. Mauch and S. Dixon, "Pyin: A fundamental frequency estimator using probabilistic threshold distributions," in *ICASSP*, 2014.

[30] M. Ott, S. Edunov, A. Baevski, A. Fan, S. Gross, N. Ng, D. Grangier, and M. Auli, "fairseq: A fast, extensible toolkit for sequence modeling," in *Proceedings of NAACL-HLT 2019: Demonstrations*, 2019.

[31] M. Mohamed, O. D. Liu, H. Tang, and S. Goldwater, "Orthogonality and isotropy of speaker and phonetic information in self-supervised speech representations," in *INTERSPEECH*, 2024.

[32] J. Yu, W. Han, A. Gulati, C.-C. Chiu, B. Li, T. N. Sainath, Y. Wu, and R. Pang, "Dual-mode asr: Unify and improve streaming asr with full-context modeling," in *ICLR*, 2020.

[33] X. Cai, D. Qiu, S. Ding, D. Hwang, W. W. A. Bruguier, R. Prabhavalkar, T. Sainath, and Y. He, "Efficient cascaded streaming asr system via frame rate reduction," in *ASRU*, 2023.

[34] T. Doutre, W. Han, M. Ma, Z. Lu, C.-C. Chiu, R. Pang, A. Narayanan, A. Misra, Y. Zhang, and L. Cao, "Improving streaming automatic speech recognition with non-streaming model distillation on unsupervised data," in *ICASSP*, 2021.

[35] S. Kumar, S. Madikeri, J. Zuluaga-Gomez, E. Villatoro-Tello, I. Thorbecke, P. Motlicek, A. Ganapathiraju *et al.*, "Xlsr-transducer: Streaming asr for self-supervised pretrained models," *arXiv preprint arXiv:2407.04439*, 2024.

[36] B. Desplanques, J. Thienpondt, and K. Demuynck, "Ecapa-tdnn: Emphasized channel attention, propagation and aggregation in tdnn based speaker verification," *INTERSPEECH*, 2020.

[37] A. Radford, J. W. Kim, T. Xu, G. Brockman, C. McLeavey, and I. Sutskever, "Robust speech recognition via large-scale weak supervision," in *ICML*, 2023.