GENERATIVE PRE-TRAINING FOR SPEECH WITH AUTOREGRESSIVE PREDICTIVE CODING

Yu-An Chung, James Glass

Computer Science and Artificial Intelligence Laboratory
Massachusetts Institute of Technology
Cambridge, MA 02139, USA
{andyyuan, glass}@mit.edu

ABSTRACT

Learning meaningful and general representations from unannotated speech that are applicable to a wide range of tasks remains challenging. In this paper we propose to use autoregressive predictive coding (APC), a recently proposed self-supervised objective, as a generative pre-training approach for learning meaningful, non-specific, and transferable speech representations. We pre-train APC on large-scale unlabeled data and conduct transfer learning experiments on three speech applications that require different information about speech characteristics to perform well: speech recognition, speech translation, and speaker identification. Extensive experiments show that APC not only outperforms surface features (e.g., log Mel spectrograms) and other popular representation learning methods on all three tasks, but is also effective at reducing downstream labeled data size and model parameters. We also investigate the use of Transformers for modeling APC and find it superior to RNNs.

Index Terms— representation learning, self-supervised learning, pre-training, transfer learning, autoregressive modeling

1. INTRODUCTION

The goal of speech representation learning is to find a transformation from surface features such as waveforms and spectrograms that makes high-level properties of speech (e.g., phonetic content, speaker characteristics, and even emotional cues) more accessible to downstream tasks. Unsupervised or self-supervised objectives are especially appealing for learning representations as they can leverage unlabeled data, which are much cheaper to obtain and more scalable than datasets requiring annotation. Representations learned via unsupervised approaches are also less likely to be biased toward a certain set of problems [1, 2, 3, 4, 5], and have more potential to be applied to a wide range of tasks.

In this paper, we aim to derive a generative pre-training approach that learns general and meaningful speech representations transferable to a variety of, potentially unknown, downstream speech tasks, where each task may require information about a different aspect of speech to perform well. For example, phonetic content may be more crucial to speech recognition, while speaker-related applications may value the speaker information more.

Due to the required generality, we argue that it is necessary to retain in the representations as much information about the original signals as possible, and let the downstream model select which information in the representations are most useful for the task it is tackling. However, most existing representation learning objectives [6, 7, 8, 9, 10] are designed to remove certain variabilities in speech (such as noise or speaker, depending on their design) and thus risk discarding information that could be useful for unknown downstream tasks. Autoregressive predictive coding (APC) [1], on the other hand, has been shown capable of learning representations that preserve information about the original signals, thus making them more *accessible* for downstream usage, where accessibility is defined as how linearly separable the representations are. This makes APC an ideal generative pre-training approach for transfer learning.

The rest of the paper is organized as follows. In Section 2 we briefly review the objective of APC and introduce two types of architectures to work as its backbone. In Section 3 we describe how we perform transfer learning with APC. Experiments and analysis on speech recognition, speech translation, and speaker identification are presented in Section 4. Finally, we conclude in Section 5 and point out some interesting future directions.

2. AUTOREGRESSIVE PREDICTIVE CODING

2.1. Objective

Autoregressive predictive coding (APC) [1] considers the sequential structures of speech and attempts to predict information about a future frame. Inspired by the neural language modeling objective for text [11], which models the likelihood of a sequence of tokens to appear as a legit language, APC is trained to understand what a reasonable spectrogram should look like and encode such information in the representations. Given a speech utterance represented as a sequence of acoustic feature vectors (e.g., log Mel spectrograms) $\mathbf{x} = (x_1, x_2, ..., x_N)$, APC incorporates an encoder Enc that encodes each frame x_i one at a time autoregressively until the current frame x_k, and tries to predict a future frame $\mathbf{x}_{k+n}$ that is n steps ahead of x_k. $n \geq 1$ is meant to encourage Enc to infer more global structures in speech rather than exploiting local smoothness of signals. At each time step, Enc produces an output prediction y_i that has the same dimensionality as x_i. Enc is optimized by minimizing the L1 loss between the predicted sequence $\mathbf{y} = (y_1, y_2, ..., y_N)$ and the target sequence $\mathbf{t} = (t_1, t_2, ..., t_N)$, which can be easily generated by right-shifting the input sequence $\mathbf{x}$ by n time steps:

$$\sum_{i=1}^{N-n} |t_i - y_i|, t_i = x_{i+n}. \tag{1}$$

Since the training target is derived from its input, APC is self-supervised and can benefit from large quantities of unlabeled data.

Code and pre-trained models are available at https://github.com/iamyuanchung/Autoregressive-Predictive-Coding.

978-1-5090-6631-5/20/$31.00 ©2020 IEEE

2.2. Encoder model

We consider two implementations of the encoder Enc for processing $\mathbf{x} = (x_1, x_2, ..., x_N)$ and producing $\mathbf{y} = (y_1, y_2, ..., y_N)$ in an autoregressive fashion: an RNN and a Transformer [12]. For the RNN, we use standard L-layer unidirectional GRUs [13]:

$$\begin{aligned} \mathbf{h}_0 &= \mathbf{x}, \\ \mathbf{h}_l &= \mathrm{GRU}^{(l)}(\mathbf{h}_{l-1}), \forall l \in [1, L], \\ \mathbf{y} &= \mathbf{W}\mathbf{h}_L, \end{aligned} \tag{2}$$

where $\mathbf{W}$ projects the output of the last RNN layer $\mathbf{h}_L$ to the dimensionality of $\mathbf{x}$. For an RNN-based APC, the set of trainable parameters is: $\{\mathbf{W}, \mathrm{GRU}^{(1)}, ..., \mathrm{GRU}^{(L)}\}$.

For the Transformer, similar to [14, 15], we consider a stack of L identical *decoder blocks* of the original architecture [12]. Each block applies a multi-headed self-attention operation over the input sequence followed by a position-wise feedforward layer for producing the input to the next block. We follow [12] and use the sinusoidal positional encodings, which do not introduce additional parameters, to provide positional information of $\mathbf{x}$ to the model:

$$\begin{aligned} \mathbf{h}_0 &= \mathbf{W}_{\mathrm{in}}\mathbf{x} + P(\mathbf{x}), \\ \mathbf{h}_l &= \mathrm{TRF}^{(l)}(\mathbf{h}_{l-1}), \forall l \in [1, L], \\ \mathbf{y} &= \mathbf{W}_{\mathrm{out}}\mathbf{h}_L, \end{aligned} \tag{3}$$

where TRF stands for Transformer, $P(\cdot)$ denotes the sinusoidal encoding function, $\mathbf{W}_{\mathrm{in}}$ is an affinity that maps $\mathbf{x}$ to the dimensionality of the Transformer hidden state, and $\mathbf{W}_{\mathrm{out}}$ is another affinity that maps the final Transformer output $\mathbf{h}_L$ back to the dimensionality of $\mathbf{x}$. The set of trainable parameters of a Transformer-based APC is: $\{\mathbf{W}_{\mathrm{in}}, \mathbf{W}_{\mathrm{out}}, \mathrm{TRF}^{(1)}, ..., \mathrm{TRF}^{(L)}\}$. In practice, we tie $\mathbf{W}_{\mathrm{in}}$ and $\mathbf{W}_{\mathrm{out}}$ by setting $\mathbf{W}_{\mathrm{in}} = \mathbf{W}_{\mathrm{out}}^T$ as a regularization.

3. TRANSFER LEARNING WITH APC

3.1. Pre-training data

We use the LibriSpeech corpus (only the speech portion) [16] for training APC. Specifically, the `train-clean-360` subset, which contains 360 hours of audio produced by 921 speakers in total, is used. We use 80-dimensional log Mel spectrograms (normalized to zero mean and unit variance per speaker) as input features. We also explore the effect of different n (Equation 1).

3.2. Transfer learning approaches

Once an APC feature extractor Enc is trained, for a downstream labeled dataset $\{(\mathbf{x}_j, c_j)\}_{j=1}^S$, where $(\mathbf{x}_j, c_j)$ is a (feature, label) pair and S denotes the training size, we transform the surface features $\{\mathbf{x}_j\}_{j=1}^S$ (in our case, the log Mel spectrograms) into a higher-level representation with Enc and obtain a new dataset $\{(\mathrm{Enc}(\mathbf{x}_j), c_j)\}_{j=1}^S$. Note that c_j can be a sequence or a single value, depending on the task.

We simply take the output of the last layer of RNN or Transformer as the extracted representations, i.e., $\mathrm{Enc}(\mathbf{x}) = \mathbf{h}_L$ in Equations 2 and 3, although there are potentially better approaches that combine the internal representations across all layers [17].

When training a downstream model with $\{(\mathrm{Enc}(\mathbf{x}_j), c_j)\}_{j=1}^S$, one possibility is to keep Enc frozen and only optimize the model; another way is to update Enc as well so that the extracted representations are better adapted to the task of interest. We examine both approaches in Section 4.

4. EXPERIMENTS

We consider three important tasks for our transfer learning experiments: (1) automatic speech recognition, (2) speaker identification, and (3) automatic speech translation. For each task, we describe the used dataset and downstream model in their respective section.

4.1. APC training details

As introduced in Section 2, we consider two architectures as the backbone of APC: RNN (Equation 2) and Transformer (Equation 3), denoted as R-APC and T-APC, respectively. For R-APC, we use 4-layer unidirectional GRUs with 512 hidden units. Following [1], we employ residual connections [18] between two consecutive layers. For T-APC, we construct a 4-layer *decoder-only* Transformer with a hidden size of 512; each layer consists of an 8-headed self-attention module followed by a 1-layer MLP with 2048 hidden units and a GELU activation function [19]. Both R-APC and T-APC are trained for 100 epochs using Adam [20] with a batch size of 32 and an initial learning rate of 10^{-3}.

4.2. Comparing methods

We compare APC with two recently proposed self-supervised representation learning objectives: contrastive predictive coding (CPC) [6] and problem-agnostic speech encoder (PASE) [3].

CPC and APC share a similar learning methodology, which is to predict information about a future frame x_{k+n} based on a history $H = (x_1, x_2, ..., x_k)$. However, instead of trying to directly predict x_{k+n} given H via regression, CPC aims to learn representations containing information that are most discriminative between x_{k+n} and a set of randomly sampled frames $\{\tilde{x}\}$. The origin distribution where $\{\tilde{x}\}$ are drawn from will largely affect what information are encoded in the representations. For example, if $\{\tilde{x}\}$ come from the same utterance as x_{k+n}, speaker information is likely to be discarded since they do not help distinguish x_{k+n} and $\{\tilde{x}\}$. Despite its effectiveness in tasks where the type of useful information is known (so one can select the sampling strategy accordingly), CPC might not be an ideal generative pre-training approach due to its lack of flexibility for learning general representations.

We mainly follow [6] for implementing CPC with some modifications described in [1]. As for APC, we also train CPC with the LibriSpeech `train-clean-360` subset.

PASE is a feature extractor trained by jointly optimizing multiple self-supervised objectives, where the learning target for each objective can be generated from the input signals. Ideally, solving each task contributes to prior knowledge into the representation, resulting in a more general one that is potentially suitable for transfer learning.

Unfortunately, we are unable to train our own PASE using `train-clean-360`, probably due to the complexity of optimizing multiple objectives simultaneously. Therefore, we directly use the pre-trained PASE model released by the authors [3]. This model was trained on about 10 hours of LibriSpeech audio—according to [3], they first aggregated all subsets in LibriSpeech, resulting in about 1,000 hours of audio produced by 2,484 speakers in total, then randomly selected utterances from the full set to exploit about 15 seconds of training material for each speaker.

For a fair comparison, we also train APC and CPC with approximately 10 hours of audio randomly selected from `train-clean-360`. Note that although they are trained on about the same amount of audio, PASE has actually seen more speakers while each speaker also has fewer training material. We add a subscript 10 to a model (e.g., CPC$_{10}$) if it is trained on only 10 hours of audio.

Below we present our transfer learning results on the three considered tasks, starting with automatic speech recognition (ASR).

4.3. Speech recognition

We conduct ASR experiments on the Wall Street Journal (WSJ) [21] corpus. We follow the standard split, using 90% of `si284` (about 72 hours) for training, the rest 10% for development, and reporting word error rates (WER) on `dev93`. The ASR model we use is a end-to-end, sequence-to-sequence (seq2seq) with attention architecture [22] composed of an encoder and a decoder. The encoder consists of 2 convolutional layers for downsampling the input features followed by a 4-layer bidirectional 256-dim GRU network. The decoder is a 1-layer unidirectional 256-dim GRU network. The seq2seq model is trained for 100 epochs using Adam with a batch size of 16 and a learning rate of 10^{-3}. For decoding, we use beam search with a beam size of 5. The baseline WER using log Mel spectrograms as input features is 18.3.

Table 1: ASR results (WER ⇓) of APC with varying n during pre-training and different transfer learning approaches (Frozen vs. Finetuned). log Mel is the baseline that uses log Mel spectrograms as input features. The best transfer learning result is marked in bold.

Features	n					
	1	2	3	5	10	20
log Mel	18.3					
R-APC Scratch	23.2					
R-APC Frozen	17.2	15.8	15.2	16.3	17.8	20.9
R-APC Finetuned	18.2	17.6	16.9	18.2	19.7	21.7
T-APC Scratch	25.0					
T-APC Frozen	19.0	16.1	14.1	**13.7**	15.4	21.3
T-APC Finetuned	22.4	17.0	15.5	14.6	16.9	23.3

The first experiment, presented in Table 1, identifies the best future time step to predict when training APC (n in Equation 1) and transfer learning approach (whether to update the pre-trained APC weights). We also include the case where APC is randomly initialized and trained from scratch along with the seq2seq model.

From Table 1 we observe that there exists a sweep spot when we vary n for both R-APC and T-APC regardless of the transfer learning approach. We think this is because for a small n, APC can exploit local smoothness in the spectrograms for predicting the target future frame (since x_k can be very similar to x_{k+n} when n is small) and thus does not need to learn to encode information useful for inferring more global structures; an overly large n, on the other hand, makes the prediction task too challenging such that APC is unable to generalize across the training set. The best n for R-APC is 3 and for T-APC it is 5. We also find that for all n, keeping pre-trained APC weights fixed (*-APC Frozen), surprisingly, works better than finetuning them (*-APC Finetuned), while the latter still outperforms the baseline. Furthermore, we see that training APC from scratch along with the seq2seq model (*-APC Scratch) always performs the worst—even worse than the baseline. With APC transfer learning, WER is reduced by more than 25% from 18.3 to 13.7.

For the rest of the experiments we adopt R-APC Frozen with $n = 3$ and T-APC Frozen with $n = 5$.

In addition to improving the performance of existing models on standard datasets, transfer learning is potentially useful for reducing the size of the downstream dataset and model needed for achieve similar performance. The intuition is that with prior knowledge, one does not need to learn automatic feature extraction from scratch. Being data-efficient is especially beneficial to low-resource languages with very few training pairs available, and a smaller model with competitive performance can mitigate the problem of having limited computational or storable resources. Below we demonstrate the effectiveness of APC transfer learning in these two aspects.

Table 2: ASR WER results with varying amounts of training data randomly sampled from `si284`. Feature extractors pre-trained with just 10 hours of LibriSpeech audio are denoted with a subscript 10.

Features	Proportion of `si284`					
	1	1/2	1/4	1/8	1/16	1/32
log Mel	18.3	24.1	33.4	44.6	66.4	87.7
CPC	20.7	28.3	38.8	50.9	69.7	88.1
R-APC	15.2	18.3	24.6	35.8	49.0	66.8
T-APC	13.7	16.4	21.3	31.4	43.0	63.2
PASE$_{10}$	20.8	26.6	32.8	42.1	58.8	78.6
CPC$_{10}$	23.4	30.0	40.1	53.5	71.3	89.3
R-APC$_{10}$	17.6	22.7	28.9	38.6	55.3	73.7
T-APC$_{10}$	18.0	23.8	31.6	43.4	61.2	80.4

In Table 2 we compare APC with other feature extractors using varying amounts of labeled data. For example, 1/16 means that we take only $72 \times 1/16 = 4.5$ hours from `si284` for training. We find that for all input features, there is a significant increase in WER whenever the training size is reduced by half. When comparing R-APC and T-APC with log Mel, we see the former two always outperform the latter across all proportions, and the gap becomes larger as training size decreases. Note that when using only half of `si284` for training, R-APC already matches the performance of log Mel trained on the full set (18.3), and T-APC even outperforms it (16.4 vs. 18.3). In particular, we observe that T-APC always outperforms log Mel by using half of the training data log Mel uses.

When comparing the bottom half (where the feature extractors are trained on just 10 hours of audio) of Table 2 with the upper part, we see that using more pre-training data is indeed helpful—performance of both CPC and APC are improved across all proportions. This observation aligns with the findings in recent NLP literature [23, 24] where having more pre-training data leads to better transfer learning results. Finally, we see that most of the time APC outperforms CPC and PASE. In some cases PASE is slightly better than T-APC$_{10}$ (e.g., when only 1/8 or less of `si284` is available), but is still worse than R-APC$_{10}$.

Table 3: ASR WER results using different numbers of GRU layers for the encoder in the ASR seq2seq model.

Features	Number of encoder layers			
	1	2	3	4
log Mel	28.8	23.5	20.8	18.3
CPC	34.3	29.8	25.2	23.7
R-APC	26.2	20.3	17.6	15.2
T-APC	25.2	18.6	15.8	13.7
PASE$_{10}$	29.4	25.7	22.5	20.8
CPC$_{10}$	35.8	31.3	26.0	24.4
R-APC$_{10}$	27.6	22.3	19.6	17.6
T-APC$_{10}$	28.1	23.2	20.6	18.0

The next aspect we examine is to what extent can we reduce the downstream model size with transfer learning. Specifically, in Table 3 we present the results of using different numbers of GRU layers $\in \{1, 2, 3, 4\}$ for constructing the encoder in the seq2seq model. We see that when using the same number of layers, *-APC and *-APC$_{10}$ always outperform other features. It is noteworthy that T-APC with just 2 layers performs similar to log Mel using 4 layers (18.6 vs. 18.3), which demonstrates the effectiveness of APC transfer learning for reducing downstream model size.

4.4. Speech translation

Our second task is automatic speech translation (AST), where the goal is to translate speech in one language into text in another. We use an English-to-French translation dataset [25] augmented from the LibriSpeech corpus [16] dedicated for this task. Each data pair consists of an English waveform and its French text translation. Following [26], we split the original training set, containing about 100 hours of audio, into 90% for training and 10% for development, and report BLEU scores [27] on the dev and test sets. The AST model we use is an RNN-based, end-to-end seq2seq with attention architecture identical to [28]. The baseline BLEU scores using log Mel as input features on the dev and test sets are 12.5 and 12.9, respectively.

For comparison, we include the performance of the cascaded system reported in [28]. The cascaded system pipelines an ASR module that first transcribes the input speech into text, and a machine translation (MT) module that translates the text to the target language. A cascaded system is usually more expensive to train than an end-to-end model as it requires intermediate audio transcriptions in the source language, but serves as a strong baseline. We also include the performance of a recently proposed Transformer-based, end-to-end AST model, dubbed S-Transformer [29], which has been shown to outperform RNN-based end-to-end model.

Table 4: Speech translation results. BLEU scores (↑) are reported. We also include the results of the cascaded system (ASR + MT) reported in [28] and the S-Transformer model reported in [29]. Only the results on the test set are available for these two approaches.

Methods	dev	test
Cascaded	-	14.6
S-Transformer	-	13.8
log Mel	12.5	12.9
CPC	12.1	12.5
R-APC	13.5	13.8
T-APC	13.7	14.3
PASE$_{10}$	12.0	12.4
CPC$_{10}$	11.8	12.3
R-APC$_{10}$	13.2	13.7
T-APC$_{10}$	12.8	13.4

From Table 4 we see that APC, regardless of how much pre-training data is used and the type of Enc, always outperforms log Mel, CPC, and PASE on both dev and test sets. Besides, our RNN-based model with T-APC features (14.3) outperforms S-Transformer (13.8), and is comparable with the cascaded system (14.6).

4.5. Speaker identification

Our final task, speaker identification (SID), examines how much transferable speaker information is captured by the representations learned by different objectives. We use WSJ for our SID experiments. We split `si284` into 80% for training, 10% for development, and 10% for testing. The task is equivalent to a 259-speaker classification problem. Features are fed into a 1-layer GRU network with a Softmax layer appended on top of the output of the last time step, which is optimized by minimizing the negative log-likelihood across the training set. We investigate settings where different amounts of utterances per speaker are used for training—in the most extreme case only one utterance per speaker is available. Exploring such one- or few-shot learning scenarios is especially interesting as it is closer to the real world where, for instance, a speech application on a personal device needs to quickly adapt to user-specific features with just a few input samples for better user experience.

Table 5: Speaker ID results. Accuracies (↑) are reported.

Features	Number of utterances per speaker seen in training					
	1	5	10	20	50	full (130 in avg.)
log Mel	8.7	43.7	60.4	70.5	87.4	96.1
CPC	13.0	45.5	65.8	75.9	89.3	96.5
R-APC	17.2	56.9	73.3	87.4	95.1	99.0
T-APC	17.6	58.6	74.4	87.8	96.3	99.1
PASE$_{10}$	12.5	48.6	64.8	79.6	92.6	96.7
CPC$_{10}$	11.7	44.9	63.2	74.6	88.3	95.8
R-APC$_{10}$	14.3	54.4	72.3	87.1	95.0	98.9
T-APC$_{10}$	13.5	49.2	70.5	82.8	92.4	98.0

From Table 5 we see that APC representations contain more transferable speaker information than all the other features, almost always outperforming them regardless of how many utterances per speaker are seen during training. It is noteworthy that T-APC is almost twice as good as log Mel (17.6 vs. 8.7) in one-shot learning.

5. CONCLUSIONS

We demonstrate that autoregressive predictive coding (APC) is an effective generative pre-training objective for transfer learning to a wide range of speech tasks. We use a Transformer to model APC and empirically show that it is more effective than an RNN used in [1]. On speech recognition (ASR), speech translation, and speaker identification (SID), representations learned by APC consistently and, mostly, significantly outperform log Mel spectrograms and representations learned by other objectives such as CPC [6] and PASE [3]. We also investigate the data efficiency and model efficiency aspects on ASR and SID, and show that APC representations are the most effective among all comparing methods at reducing downstream labeled data size and model parameters.

There are many interesting directions for future work. In our experiments, we find that keeping APC weights frozen works better than updating them when training on downstream dataset. However, we believe that the latter is more ideal for transfer learning as it adapts the extracted representations toward the target task. More sophisticated techniques for fine-tuning [30] could be used. Regarding the backbone architecture of APC, the Transformer model can be potentially improved by modifying the way we inject positional information [31, 32]. Additionally, as hinted by recent NLP research [23, 24], training APC on more unlabeled data is also a promising way for improving the transfer learning results. Finally, we are interested in exploring the usage of APC in other speech applications such as speech synthesis, where pre-training and transfer learning have already achieved some success [33, 34, 35, 36, 37].

6. REFERENCES

[1] Yu-An Chung, Wei-Ning Hsu, Hao Tang, and James Glass, "An unsupervised autoregressive model for speech representation learning," in *Interspeech*, 2019.

[2] Jan Chorowski, Ron Weiss, Samy Bengio, and Aäron van den Oord, "Unsupervised speech representation learning using wavenet autoencoders," *IEEE/ACM TASLP*, vol. 27, no. 12, pp. 2041–2053, 2019.

[3] Santiago Pascual, Mirco Ravanelli, Joan Serrà, Antonio Bonafonte, and Yoshua Bengio, "Learning problem-agnostic speech representations from multiple self-supervised tasks," in *Interspeech*, 2019.

[4] Steffen Schneider, Alexei Baevski, Ronan Collobert, and Michael Auli, "Wav2vec: Unsupervised pre-training for speech recognition," in *Interspeech*, 2019.

[5] Herman Kamper, "Truly unsupervised acoustic word embeddings using weak top-down constraints in encoder-decoder models," in *ICASSP*, 2019.

[6] Aaron van den Oord, Yazhe Li, and Oriol Vinyals, "Representation learning with contrastive predictive coding," *arXiv preprint arXiv:1807.03748*, 2018.

[7] Benjamin Milde and Chris Biemann, "Unspeech: Unsupervised speech context embeddings," in *Interspeech*, 2018.

[8] Yu-An Chung and James Glass, "Speech2vec: A sequence-to-sequence framework for learning word embeddings from speech," in *Interspeech*, 2018.

[9] Yi-Chen Chen, Sung-Feng Huang, Chia-Hao Shen, Hung-Yi Lee, and Lin-Shan Lee, "Phonetic-and-semantic embedding of spoken words with applications in spoken content retrieval," in *SLT*, 2018.

[10] Wei-Ning Hsu, Yu Zhang, and James Glass, "Unsupervised learning of disentangled and interpretable representations from sequential data," in *NIPS*, 2017.

[11] Tomáš Mikolov, Martin Karafiát, Lukáš Burget, Jan Černockỳ, and Sanjeev Khudanpur, "Recurrent neural network based language model," in *Interspeech*, 2010.

[12] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, et al., "Attention is all you need," in *NIPS*, 2017.

[13] Kyunghyun Cho, Bart van Merrienboer, Dzmitry Bahdanau, and Yoshua Bengio, "On the properties of neural machine translation: Encoder-decoder approaches," in *SSST*, 2014.

[14] Peter Liu, Mohammad Saleh, Etienne Pot, Ben Goodrich, Ryan Sepassi, et al., "Generating wikipedia by summarizing long sequences," in *ICLR*, 2018.

[15] Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever, "Improving language understanding by generative pre-training," Tech. Rep., OpenAI, 2018.

[16] Vassil Panayotov, Guoguo Chen, Daniel Povey, and Sanjeev Khudanpur, "Librispeech: An ASR corpus based on public domain audio books," in *ICASSP*, 2015.

[17] Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer, "Deep contextualized word representations," in *NAACL-HLT*, 2018.

[18] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun, "Deep residual learning for image recognition," in *CVPR*, 2016.

[19] Dan Hendrycks and Kevin Gimpel, "Gaussian error linear units (gelus)," *arXiv preprint arXiv:1606.08415*, 2016.

[20] Diederik Kingma and Jimmy Ba, "Adam: A method for stochastic optimization," in *ICLR*, 2015.

[21] Douglas Paul and Janet Baker, "The design for the wall street journal-based CSR corpus," in *Speech and Natural Language Workshop*, 1992.

[22] Jan Chorowski, Dzmitry Bahdanau, Dmitriy Serdyuk, Kyunghyun Cho, and Yoshua Bengio, "Attention-based models for speech recognition," in *NIPS*, 2015.

[23] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova, "BERT: Pre-training of deep bidirectional Transformers for language understanding," in *NAACL-HLT*, 2019.

[24] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, et al., "RoBERTa: A robustly optimized BERT pretraining approach," *arXiv preprint arXiv:1907.11692*, 2019.

[25] Ali Kocabiyikoglu, Laurent Besacier, and Olivier Kraif, "Augmenting Librispeech with French translations: A multimodal corpus for direct speech translation evaluation," in *LREC*, 2018.

[26] Yu-An Chung, Wei-Hung Weng, Schrasing Tong, and James Glass, "Towards unsupervised speech-to-text translation," in *ICASSP*, 2019.

[27] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu, "BLEU: A method for automatic evaluation of machine translation," in *ACL*, 2002.

[28] Alexandre Bérard, Laurent Besacier, Ali Can Kocabiyikoglu, and Olivier Pietquin, "End-to-end automatic speech translation of audiobooks," in *ICASSP*, 2018.

[29] Mattia Di Gangi, Matteo Negri, and Marco Turchi, "Adapting Transformer to end-to-end spoken language translation," in *Interspeech*, 2019.

[30] Jeremy Howard and Sebastian Ruder, "Universal language model fine-tuning for text classification," in *ACL*, 2018.

[31] Abdelrahman Mohamed, Dmytro Okhonko, and Luke Zettlemoyer, "Transformers with convolutional context for ASR," *arXiv preprint arXiv:1904.11660*, 2019.

[32] Kazuki Irie, Albert Zeyer, Ralf Schlüter, and Hermann Ney, "Language modeling with deep Transformers," in *Interspeech*, 2019.

[33] Yu-An Chung, Yuxuan Wang, Wei-Ning Hsu, Yu Zhang, and RJ Skerry-Ryan, "Semi-supervised training for improving data efficiency in end-to-end speech synthesis," in *ICASSP*, 2019.

[34] Yuan-Jui Chen, Tao Tu, Cheng-Chieh Yeh, and Hung-Yi Lee, "End-to-end text-to-speech for low-resource languages by cross-lingual transfer learning," in *Interspeech*, 2019.

[35] Tomoki Hayashi, Shinji Watanabe, Tomoki Toda, Kazuya Takeda, Shubham Toshniwal, and Karen Livescu, "Pre-trained text embeddings for enhanced text-to-speech synthesis," in *Interspeech*, 2019.

[36] Wei Fang, Yu-An Chung, and James Glass, "Towards transfer learning for end-to-end speech synthesis from deep pre-trained language models," Tech. Rep., Massachusetts Institute of Technology, 2019.

[37] Ye Jia, Yu Zhang, Ron Weiss, Quan Wang, Jonathan Shen, et al., "Transfer learning from speaker verification to multi-speaker text-to-speech synthesis," in *NeurIPS*, 2018.