# EVALUATING OFF-THE-SHELF MACHINE LISTENING AND NATURAL LANGUAGE MODELS FOR AUTOMATED AUDIO CAPTIONING

*Benno Weck*[1,2]*, Xavier Favory*[2]*, Konstantinos Drossos*[3]*, Xavier Serra*[2]

[1] Huawei Technologies, Munich Research Center, Germany
{firstname.lastname}@huawei.com
[2] Music Technology Group, Universitat Pompeu Fabra, Spain
benno.weck01@estudiant.upf.edu,{firstname.lastname}@upf.edu
[3] Audio Research Group, Tampere University, Finland
{firstname.lastname}@tuni.fi

## ABSTRACT

Automated audio captioning (AAC) is the task of automatically generating textual descriptions for general audio signals. A captioning system has to identify various information from the input signal and express it with natural language. Existing works mainly focus on investigating new methods and try to improve their performance measured on existing datasets. Having attracted attention only recently, very few works on AAC study the performance of existing pre-trained audio and natural language processing resources. In this paper, we evaluate the performance of off-the-shelf models with a Transformer-based captioning approach. We utilize the freely available Clotho dataset to compare four different pre-trained machine listening models, four word embedding models, and their combinations in many different settings. Our evaluation suggests that YAMNet combined with BERT embeddings produces the best captions. Moreover, in general, fine-tuning pre-trained word embeddings can lead to better performance. Finally, we show that sequences of audio embeddings can be processed using a Transformer encoder to produce higher-quality captions.

***Index Terms—*** audio captioning, transfer learning, word embeddings, machine listening, transformer

## 1. INTRODUCTION

Automated audio captioning (AAC) is an inter-modal translation task, where existing methods take an audio signal as input and generate a textual description, i.e. a caption, of its contents [1]. The generated captions contain information about various aspects of the content of the audio signal, ranging from identification of sound events to knowledge about spatiotemporal interactions, foreground and background disambiguation, surroundings, textures, and other high-level information [2–4].

To our knowledge, all published works focusing on AAC solely employ deep learning methods [1–3, 5–17]. Most of them follow an encoder-decoder scheme and address the task as a sequence-to-sequence (seq2seq) learning problem [18]. Convolutional neural network (CNN)-based encoders are often utilized, for example, in [2, 3, 7, 8, 12, 17], and recurrent neural network (RNN)-based decoders can be used in order to generate the captions [1–3, 7, 9, 11–17]. Recently, more and more methods have involved an attention mechanism. For example, as a technique to enable the decoder to focus only on certain parts of the latent representation extracted by the encoder [1, 2, 17]. Or more generally, other approaches [6, 8, 10, 19, 20] employ a Transformer model [21]. This type of model seems particularly adequate for AAC since it led to groundbreaking results in multiple fields, such as natural language processing (NLP), computer vision, and audio processing [22].

Transfer learning is a popular technique often employed in NLP and machine listening (MaL) tasks. However, existing approaches in AAC often do not take advantage of any pre-trained resources and instead train their models from scratch. Only recently, a few published papers [2,3,5,6] propose to incorporate pre-trained audio models such as VGGish [23] or to rely on word embedding models such as word2vec [24]. Given the large number of available pre-trained models in MaL and NLP, it is still unclear which models are most suited for AAC. Moreover, incorporating these models into a Transformer-based AAC system can involve some specific design choices that are also yet overlooked.

In this paper, we focus on investigating the use of pre-trained models taken from MaL and NLP in the context of AAC. In particular, we are interested in identifying which available resources are the most valuable and how to combine them efficiently in a Transformer-based AAC system. We use various off-the-shelf pre-trained audio and word encoding methods. Our contributions are:

- We adapt a Transformer-based AAC method that can use different pre-trained MaL and NLP models,
- We conduct a thorough investigation of the performance of our method by combining pre-trained models in various settings,
- We identify what combinations of techniques make pre-trained resources specifically beneficial for an AAC system. We consider fine-tuning word embeddings, using an adapter to process audio embeddings, and the usage of overlap when extracting audio embeddings.

The rest of the paper is structured as follows: In Section 2 we present our method and in Section 3 we outline the evaluation process. The results are presented and discussed in Section 4. Finally, Section 5 concludes the paper.

## 2. METHOD

In our study, we adopt a Transformer-based model architecture which has been shown to produce state-of-the-art results for AAC [10, 19, 20]. An overview of our method is presented in

Figure 1. It consists of an audio encoder, E(·), an embeddings' adapter, A(·), and a decoder D(·). As E, we employ different pre-trained models for general audio processing. For A, we compare the use of no adapter, a multi-layer perceptron (MLP), and a multi-head attention (MHA) component. The output of A is used together with word embeddings of the previously predicted words as an input to D, a Transformer-based decoder[1].

A sequence of audio features $\mathbf{X} \in \mathbb{R}^{T \times F}$ with $T$ vectors of $F$ features is used as an input to the method, which outputs a sequence of one-hot encoded tokens $\mathbf{S} \in [0,1]^{K \times W}$, where $K$ corresponds to the number of tokens in the generated caption and $W$ to the size of the considered vocabulary. More specifically, $\mathbf{X}$ is used as an input to E as

$$\mathbf{Z} = \mathrm{E}(\mathbf{X}), \tag{1}$$

where $\mathbf{Z} \in \mathbb{R}^{T' \times F'}$ is a sequence of $T'$ intermediate representations with $F'$ features provided by the pre-trained model (i.e. an audio embedding sequence). Then, the adapter A will process $\mathbf{Z}$ as

$$\mathbf{Z}' = \mathrm{A}(\mathbf{Z}), \tag{2}$$

where $\mathbf{Z}' \in \mathbb{R}^{T' \times F''}$ and $F''$ is the dimensionality of the features that A outputs. Finally, the decoder D will predict the probability distribution of appearance over the $W$ words at the $k$-th step, $\mathbf{S}_k$, as

$$\mathbf{S}_k = \mathrm{D}(\mathbf{Z}', \mathbf{S}'_0, \ldots, \mathbf{S}'_{k-1}), \tag{3}$$

where $\mathbf{S}'_i$ is a learned word embedding for step $i$, and $\mathbf{S}'_0 = \{0\}^{W'}$. As $\mathbf{S}'$, we make use of different pre-trained NLP models.

We employ different audio embedding models that are optimized for a task different from AAC. The extracted audio embeddings might contain information that is specific to the corresponding source task and not necessarily optimal for AAC. We do not fine-tune the models in our experiment, but instead, we study the usage of different adapters A that process the audio embeddings $\mathbf{Z}$.

The Transformer decoder D consists of $N$ blocks, each of them having two serially cascaded MHA layers that perform self and cross-modal (i.e. between audio and words) attention, respectively. The output of the second, cross-modal attention, is given as an input to a linear layer and a layer normalization process. The word embeddings $\mathbf{S}'$ are used as an input to D. Following the original proposal of the Transformer model, we apply a positional encoding to the input word embeddings. To generate the captions, the decoder can be sampled until the desired caption length is met or a special token indicating the end of a sentence is produced.

## 3. EVALUATION

In our study, we compare the performance of four pre-trained audio processing models, two audio embedding adapters, and four pre-trained NLP models for AAC, using the AAC dataset Clotho [25].

### 3.1. Dataset, metrics, and experiments

The Clotho dataset contains a training, a validation, and an evaluation split, comprising 3839, 1045, and 1045 audio examples, respectively. Each audio example is annotated with five captions, and we consider one audio-caption pair a single training example. The performance is assessed using the SPIDEr score [26]. This metric is widely established in the community (e.g. in the AAC task from the 2021 DCASE Challenge[2]) as it highly correlates with the human judgment of caption quality [26].

[1]For a complete description of the Transformer decoder, refer to [21].

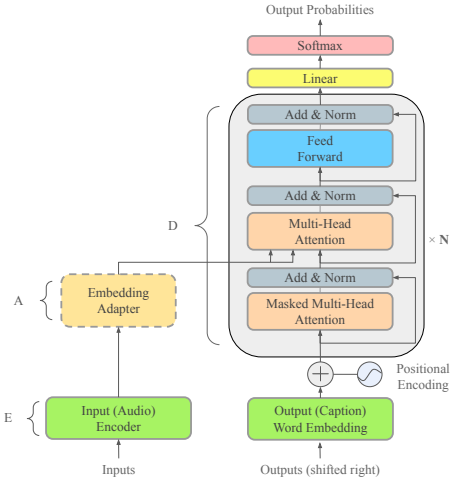


Figure 1: Model architecture.

In all our experiments, the decoder part of our model consists of $N = 3$ stacked Transformer decoder blocks with four 128-dimensional attention heads each, similar to what is used in [10,19]. During training, all models are optimized by minimizing the cross-entropy loss between the predicted sentence and the target caption using the Adam algorithm ($\alpha = 0.001, \beta_1 = 0.9, \beta_2 = 0.999$, and $\epsilon = 10^{-8}$) with a minibatch size of 256. Early stopping is applied after ten epochs with no improvement of the loss calculated on the held-out validation set. The best model according to this loss is then evaluated on the evaluation set.

To avoid bias in the results, we repeat all experiments ten times with different random initialization and report the mean statistic for all scores. We test all combinations of encoder models with different overlaps, audio embedding adapter layers, fixed or fine-tuned word embeddings. In total, it constitutes 264 different settings.

### 3.2. Audio embedding models

For each employed audio embedding model, we follow the authors' methodology to extract audio embeddings using their pre-trained models. We use the pre-trained models as encoders with frozen weights, i.e. without fine-tuning.

Existing AAC approaches use audio encoders with different hop-sizes. In this work, we study the impact of using overlap when extracting the audio embeddings. More specifically, we use two different settings for the embedding extraction hop-size, corresponding to 50% overlap and no overlap.

Table 1: Audio encoder models compared in this study.

| Encoder | Dimensionality | Window | Learning |
|---|---|---|---|
| VGGish | 128 | 0.96 s | supervised |
| YAMNet | 1024 | 0.96 s | supervised |
| OpenL3 | 512 | 1.00 s | self-supervised |
| COALA | 1152 | 2.20 s | contrastive |

[2]http://dcase.community/challenge2021/task-automatic-audio-captioning

Table 1 gives an overview of the audio source models we compared. All four are CNN-based models. The first, **VGGish** [23], is inspired by the VGG architecture mainly used in computer vision [27]. It was trained on a preliminary version of the YouTube-8M dataset in a supervised fashion [28]. It extracts 128-dimensional embeddings from roughly 1 second of audio. The second audio embedding model is **YAMNet**, which also draws inspiration from computer vision models [23]. It employs a MobileNet [29] architecture to extract embeddings with dimensionality 1024 from almost 1 second of audio. The model was trained to predict 521 audio event classes on the AudioSet dataset [30]. The third model, **OpenL3** [31], is a modified and freely available version of the $L^3$-Net [32]. OpenL3 is trained in a self-supervised way in an audio-visual correspondence task, relying on videos from AudioSet. From the multiple variants that the authors provide, we chose the model configuration that produces an embedding of size 512, and that was trained with 128 Mel bands as input representation in the environmental sound setting. The fourth and final model is **COALA** [33], a model trained by taking advantage of user-provided tags in Freesound[3] [34]. During training, it employed a contrastive learning approach to align audio and associated tag embeddings, producing an audio embedding model that can extract semantically enriched audio representations. The model produces embeddings from 2.2-second patches.

### 3.3. Adapter Layers

We compare two different adapter architectures that are depicted in Figure 2 and contrast them with applying no adapter, which we refer to as the identity function. The aim is to investigate if one kind of adapter can improve the performance of a Transformer-based AAC method. Moreover, the adapters ensure a match in dimension between the audio embeddings and the internal dimension of the decoder. It enables us to compare embeddings of different sizes — from different models — with a fixed number of decoder parameters. When not using any adapter, the first decoder layer changes in size depending on the embeddings' dimensionality.

As the first adapter, we employ a two-layer **MLP** with a hidden layer of size 256 and rectified linear unit (ReLU) as activation function, as shown in Figure 2, to compute the adapted representation with weights shared across time. The second adapter layer is an **MHA** block, followed by a linear layer and a layer normalization process, also known as a Transformer encoder layer [21]. This type of network was previously combined with a VGGish embedding model in the context of AAC [5]. It complements our decoder in such a way that our model architecture is similar to a full Transformer model for AAC [6]. The MHA block employs four attention heads of 128 dimensions. A linear dimensionality reduction function as described in [6] and a positional encoding precede it.

### 3.4. Word embedding models

Our first word embedding model is **word2vec** [24], which is based on the skip-gram algorithm. We use the publicly available model pre-trained on three million words and phrases from Google News.

Our second model is **GloVe** [35], which takes a different approach by learning context information from corpus-level word-word co-occurrence statistics rather than local context windows. The authors show that GloVe is an improvement over the word2vec algorithm in downstream word analogy and Named Entity Recognition tasks. We employ the publicly available model trained on the combination of a 2014 Wikipedia copy and the Gigaword 5 corpus [36], which together contain six billion tokens.

[3]The training data from COALA and Clotho are disjoint sets.

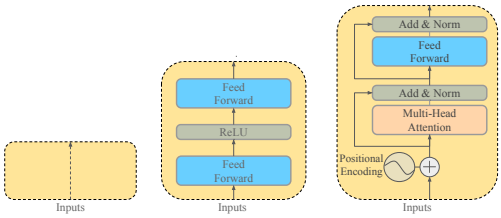


Figure 2: Embedding adapter architectures: Identity (left), MLP (middle), and MHA-based (right).

The third word embedding model is **fastText**, which implements several optimizations on top of the word2vec skip-gram algorithm [37]. FastText provides better handling of multi-word phrases, uses a weighted context, and considers subwords (i.e. character n-grams). We use the publicly available model trained with subword information on the Common Crawl corpus, which contains 600B tokens and is significantly larger than the corpora used for the Glove and word2vec model [38].

We employ **BERT** as our fourth model, which is a large language model based on the Transformer architecture and can be used as a feature extractor to extract word embeddings [39]. In contrast to models mentioned above, such as word2vec, BERT also takes the context of a token — the entire sentence — into account when extracting embeddings, i.e. producing embeddings that are context-sensitive. We use the $\text{BERT}_{\text{BASE}}$ configuration pre-trained on a Wikipedia copy (2.5B words) and the BookCorpus dataset (800M words) [40]. Different ways to use the layers of BERT as word embeddings have been discussed in the literature, and it is not clear what the best choice is for AAC. We decided to use the penultimate layer as embeddings as this can produce highly contextualized representations that are not too task-specific [41]. We extract the word embeddings from an entire caption. Due to the computational cost of the model, it will not be fine-tuned in our experiments.

Additionally, to explore if pre-trained word embeddings can be helpful, we also adopt randomly initialized word vectors and a continuous bag-of-words (CBOW) word2vec model [24] trained using the text of the captions in the Clotho training set. Finally, all word embedding models produce embeddings with $W' = 300$ dimensions, except for those from the BERT model with $W' = 768$.

## 4. RESULTS AND DISCUSSION

In this section, we show and discuss the results of our experiment. We organize our discussion around, first, the performance of the different pre-trained audio encoder models. Second, we discuss the usage of the different audio embedding adapter components. Third, we study the performance of different word embedding models and the impact of fine-tuning them. Finally, we show the potential of computing audio embeddings on overlapped audio frames.

Table 2 lists the optimal settings for each of the audio encoder models that we found, and Figure 3 displays box plots of the audio encoder models' performance for each adapter. The top-performing model in the best overall setting (YAMNet, BERT, MHA) achieved a SPIDEr score of 0.1914. Moreover, we found that YAMNet consistently outperforms the other audio models. Overall, audio en-

Table 2: Top-performing settings for pre-trained encoder models and their SPIDEr score when embeddings are extracted with no or 50% overlap ($\star$, and † respectively).

| Encoder | Word embedding | Adapter | SPIDEr Mean | SPIDEr SD |
|---|---|---|---|---|
| COALA† | BERT | MHA-based | 0.1495 | 0.0044 |
| OpenL3$^\star$ | BERT | MHA-based | 0.1620 | 0.0051 |
| VGGish$^\star$ | BERT | MHA-based | 0.1677 | 0.0052 |
| YAMNet† | BERT | MHA-based | 0.1793 | 0.0066 |

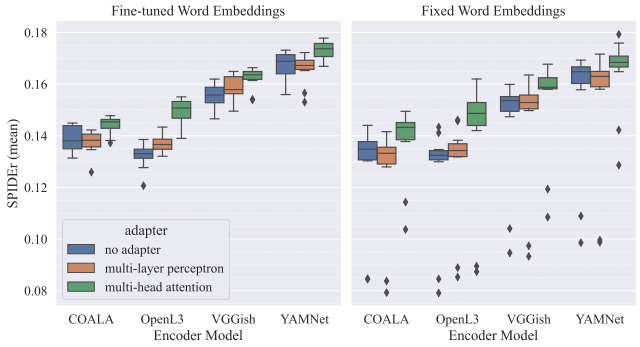


Figure 3: Comparison of SPIDEr score for different encoder models averaged over multiple different experiment settings.

coder models trained in a supervised classification task using large datasets, YAMNet & VGGish, are superior to the models that are trained in a self-supervised way or using contrastive learning. This highlights the potential of using large datasets for pre-training audio encoders in auto-tagging tasks and using them for AAC.

Using an MHA-based encoder as an adapter on top of the audio embeddings consistently provides the best results (Figure 3 and Table 2), whereas using the MLP does not provide any improvement in comparison with no adapter. Interestingly, the benefit of the MHA-based adapter is most prominent for OpenL3, which has been trained in a self-supervised way. This suggests that the audio embeddings extracted with OpenL3 contain some semantics useful for AAC that can be exploited using an adapter such as the MHA-based one. Our results suggest that employing a Transformer-based encoder using positional encoding and MHA can process sequences of audio embeddings, leading to better performance in AAC, which aligns well with findings from previous works [6, 10].

The left part of Figure 4 reports a performance comparison of the different word embeddings employed in our evaluation. On average, using the pre-trained BERT model to extract the word embeddings leads to the best performance. It is worth mentioning that we are using BERT as a fixed external word embedding model instead of using its full capacity, for example, by fine-tuning it for AAC. However, the latter would require much more computational resources (the BERT model has around 110M parameters).

Training word embedding representations from scratch during AAC provides already promising results. By using pre-trained word embeddings, such as word2vec, GloVe, and fastText, we can slightly improve this performance. Additionally, fine-tuning them can significantly improve their performance (one-sided Wilcoxon signed-rank tests $p < 0.001$, for each pre-trained word embedding model). Interestingly, optimizing the randomly initialized word

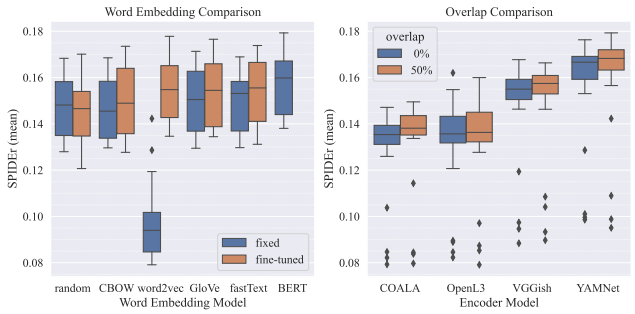


Figure 4: SPIDEr sores for word embedding (left) and audio embedding models (right). Scores are averaged over all combinations is each case.

representations does not improve their performance. This highlights the need for pre-trained word representations in the context of our AAC task.

The right side of Figure 4 displays the performance of the audio models with and without overlap when extracting the embeddings. In particular, we observe that computing the embeddings with 50% overlap leads to improved performance with two audio encoders. One-sided Wilcoxon signed-rank tests indicated that this improvement is significant for COALA ($W = 84, p < 8.08e^{-07}$) and YAMNet ($W = 112, p < 3.92e^{-06}$).

## 5. CONCLUSION

In this paper, we conduct a comparative analysis of many off-the-shelf resources from natural language processing (NLP) and the machine listening field for automated audio captioning (AAC). The core components of our method are a fixed audio encoder, an audio embedding adapter, and a Transformer-based decoder. Our results show that YAMNet outclasses the other audio embedding models when used as an encoder. The performance can be increased for two encoders (COALA & YAMNet) by computing the embeddings on overlapped frames. Processing the audio embeddings with a multi-head attention-based adapter can increase the performance of our captioning system while using a multi-layer perceptron is not different from not using any adapter. We found that pre-trained word embedding models are a valuable resource for AAC, particularly so when fine-tuned during the training. Using BERT as a fixed embedding extraction model gave the best results. This result motivates the usage of large pre-trained NLP models such as BERT to create better AAC methods.

Future work could investigate the impact of the positional encoding in the audio adapter by independently evaluating the multi-head attention adapter. Finally, fine-tuning the audio embedding models has not been studied in this work. However, it may be an essential technique that can benefit AAC approaches, as highlighted by the fact that adding an adaptation model to process the embeddings significantly increases the performance of our system.

## 6. ACKNOWLEDGMENT

K. Drossos has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 957337, project MARVEL.

## 7. REFERENCES


[1] K. Drossos, S. Adavanne, *et al.*, "Automated audio captioning with recurrent neural networks," in *2017 IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA)*, Oct. 2017, pp. 374–378.

[2] X. Xu, H. Dinkel, *et al.*, "Investigating local and global information for automated audio captioning with transfer learning," in *ICASSP 2021 - 2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2021, pp. 905–909.

[3] A. Ö. Eren and M. Sert, "Audio captioning based on combined audio and semantic embeddings," in *2020 IEEE International Symposium on Multimedia (ISM)*, 2020, pp. 41–48.

[4] S. Lipping, K. Drossos, *et al.*, "Crowdsourcing a dataset of audio captions," in *Proceedings of the Detection and Classification of Acoustic Scenes and Events 2019 Workshop (DCASE2019)*, New York University, NY, USA, October 2019, pp. 139–143.

[5] Y. Koizumi, Y. Ohishi, *et al.*, "Audio captioning using pre-trained large-scale language model guided by audio-based similar caption retrieval," *arXiv preprint arXiv:2012.07331*, 2020.

[6] Y. Koizumi, R. Masumura, *et al.*, "A transformer-based audio captioning model with keyword estimation," in *Interspeech 2020, 21st Annual Conference of the International Speech Communication Association*. Shanghai, China: ISCA, Oct. 2020, pp. 1977–1981.

[7] A. Ö. Eren and M. Sert, "Audio captioning using gated recurrent units," *arXiv*, vol. abs/2006.03391, 2020.

[8] K. Chen, Y. Wu, *et al.*, "Audio captioning based on transformer and pre-trained CNN," in *Proceedings of the Detection and Classification of Acoustic Scenes and Events 2020 Workshop (DCASE2020)*, Tokyo, Japan, Nov. 2020, pp. 21–25.

[9] K. Nguyen, K. Drossos, *et al.*, "Temporal sub-sampling of audio feature sequences for automated audio captioning," *arXiv*, vol. abs/2007.02676, 2020.

[10] A. Tran, K. Drossos, *et al.*, "Wavetransformer: An architecture for audio captioning based on learning temporal and time-frequency information," in *European Signal Processing Conference (EUSIPCO)*, Aug. 2021.

[11] E. Çakir, K. Drossos, *et al.*, "Multi-task regularization based on infrequent classes for audio captioning," *arXiv*, vol. abs/2007.04660, 2020.

[12] X. Xu, H. Dinkel, *et al.*, "A CRNN-GRU based reinforcement learning approach to audio captioning," in *Proceedings of the Detection and Classification of Acoustic Scenes and Events 2020 Workshop (DCASE2020)*, Tokyo, Japan, Nov. 2020, pp. 225–229.

[13] D. Takeuchi, Y. Koizumi, *et al.*, "Effects of word-frequency based pre- and post- processings for audio captioning," in *Proceedings of the Detection and Classification of Acoustic Scenes and Events 2020 Workshop (DCASE2020)*, Tokyo, Japan, Nov. 2020, pp. 190–194.

[14] M. Wu, H. Dinkel, *et al.*, "Audio caption: Listen and tell," in *ICASSP 2019 - 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2019, pp. 830–834.

[15] S. Ikawa and K. Kashino, "Neural audio captioning based on conditional sequence-to-sequence model," in *Proceedings of the Detection and Classification of Acoustic Scenes and Events 2019 Workshop (DCASE2019)*, New York University, NY, USA, Oct. 2019, pp. 99–103.

[16] X. Xu, H. Dinkel, *et al.*, "Audio caption in a car setting with a sentence-level loss," in *12th International Symposium on Chinese Spoken Language Processing (ISCSLP)*, Hong Kong, 2021, pp. 1–5.

[17] C. D. Kim, B. Kim, *et al.*, "Audiocaps: Generating captions for audios in the wild," in *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT*, Minneapolis, MN, USA, June 2019, pp. 119–132.

[18] I. Sutskever, O. Vinyals, *et al.*, "Sequence to sequence learning with neural networks," in *Advances in Neural Information Processing Systems*, vol. 27. Curran Associates, Inc., 2014.

[19] Y. Wu, K. Chen, *et al.*, "Audio captioning based on transformer and pre-training for 2020 DCASE audio captioning challenge," DCASE2020 Challenge, Tech. Rep., June 2020.

[20] W. Yuan, Q. Han, *et al.*, "The DCASE 2021 challenge task 6 system: Automated audio captioning with weakly supervised pre-traing and word selection methods," DCASE2021 Challenge, Tech. Rep., 2021.

[21] A. Vaswani, N. M. Shazeer, *et al.*, "Attention is all you need," *arXiv*, vol. abs/1706.03762, 2017.

[22] T. Lin, Y. Wang, *et al.*, "A survey of transformers," *arXiv*, vol. abs/2106.04554, 2021.

[23] S. Hershey, S. Chaudhuri, *et al.*, "CNN architectures for large-scale audio classification," in *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2017, pp. 131–135.

[24] T. Mikolov, I. Sutskever, *et al.*, "Distributed representations of words and phrases and their compositionality," in *NIPS*, 2013.

[25] K. Drossos, S. Lipping, *et al.*, "Clotho: an audio captioning dataset," *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 736–740, 2020.

[26] S. Liu, Z. Zhu, *et al.*, "Improved image captioning via policy gradient optimization of spider," *2017 IEEE International Conference on Computer Vision (ICCV)*, pp. 873–881, 2017.

[27] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *CoRR*, vol. abs/1409.1556, 2015.

[28] S. Abu-El-Haija, N. Kothari, *et al.*, "Youtube-8m: A large-scale video classification benchmark," *arXiv preprint arXiv:1609.08675*, 2016.

[29] A. G. Howard, M. Zhu, *et al.*, "Mobilenets: Efficient convolutional neural networks for mobile vision applications," *arXiv*, vol. abs/1704.04861, 2017.

[30] J. F. Gemmeke, D. P. W. Ellis, *et al.*, "Audio set: An ontology and human-labeled dataset for audio events," in *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2017, pp. 776–780.

[31] J. Cramer, H.-H. Wu, *et al.*, "Look, listen, and learn more: Design choices for deep audio embeddings," in *ICASSP 2019 - 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2019, pp. 3852–3856.

[32] R. Arandjelovic and A. Zisserman, "Look, listen and learn," in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 609–617.

[33] X. Favory, K. Drossos, *et al.*, "COALA: Co-aligned autoencoders for learning semantically enriched audio representations," in *Workshop on Self-supervision in Audio and Speech at the 37th International Conference on Machine Learning*, Vienna, Austria, 2020.

[34] F. Font, G. Roma, *et al.*, "Freesound technical demo," in *Proceedings of the 21st ACM International Conference on Multimedia*, ser. MM '13. New York, NY, USA: Association for Computing Machinery, 2013, p. 411–412.

[35] J. Pennington, R. Socher, *et al.*, "Glove: Global vectors for word representation," in *Empirical Methods in Natural Language Processing (EMNLP)*, 2014, pp. 1532–1543.

[36] https://catalog.ldc.upenn.edu/LDC2011T07.

[37] T. Mikolov, E. Grave, *et al.*, "Advances in pre-training distributed word representations," in *Proceedings of the International Conference on Language Resources and Evaluation (LREC 2018)*, 2018.

[38] http://commoncrawl.org/2017/06/.

[39] J. Devlin, M.-W. Chang, *et al.*, "BERT: Pre-training of deep bidirectional transformers for language understanding," in *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*. Minneapolis, Minnesota: Association for Computational Linguistics, June 2019, pp. 4171–4186.

[40] Y. Zhu, R. Kiros, *et al.*, "Aligning books and movies: Towards story-like visual explanations by watching movies and reading books," in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 19–27.

[41] A. Rogers, O. Kovaleva, *et al.*, "A primer in bertology: What we know about how BERT works," *Transactions of the Association for Computational Linguistics*, vol. 8, pp. 842–866, 2021.