TRANSFER LEARNING FOLLOWED BY TRANSFORMER FOR AUTOMATED AUDIO CAPTIONING

Hyejin Won[1], *Baekseung Kim*[1], *Il-Youp Kwak*[1*], *Changwon Lim*[1†]

[1] Chung-Ang University, Department of Applied Statistics, Seoul, South Korea,
{whj9492, kbs778, ikwak2, clim}@cau.ac.kr

ABSTRACT

Automated Audio Captioning (AAC) automatically creates captions that can explain the given audio sound data using machine learning techniques. Researchers investigate the solutions for AAC on DCASE 2021 audio captioning challenge. In the challenge, a model is required to generate natural language descriptions of a given audio signal. We use pre-trained models trained using AudioSet data, a large-scale dataset of manually annotated audio events. A large amount of audio events data would help capturing important audio feature representation. To use the learned feature from AudioSet data, we utilize CNN14 or ResNet54 network pre-trained on AudioSet, which achieved state-of-the-art audio pattern recognition performance. Our proposed sequence-to-sequence model consists of a CNN14 or ResNet54 encoder and a Transformer decoder. Experiments show that the proposed model can achieve a SPIDEr score of 0.246 and 0.285 on audio captioning performance. We further experiment the use of three different voice features, log-mel spectrogram, constant Q transform spectrogram, and gammatone filter spectrogram.

Index Terms— Automated audio captioning, Acoustic event detection, Transfer learning, Transformer

1. INTRODUCTION

Automated Audio Captioning (AAC) automatically creates captions that can explain the given audio sound data using machine learning techniques. Many researchers investigated this exciting problem from DCASE challenges and workshops in 2020 and 2021 [1, 2, 3, 4, 5, 6]. One example of a generated caption on a given sound could be "people talking in a small and empty room." The 2021 DCASE AAC uses the Clotho v2.1 dataset [7]. Clotho v2.1 data contains 6,974 (4,981 from version 1 and 1,993 from version 2.1) audio clips in 15-30 seconds each with 5 captions in 8-20 English words.

In the DCASE 2020 competition on AAC, Drossos et al. (2017) [1] introduce an encoder-decoder structured baseline for the AAC task. The encoder-decoder structure is the most widely used architecture for AAC. The baseline model has an encoder-decoder scheme with a multi-layered, bi-directional GRU encoder and multi-layered decoder. Takeuchi et al. (2020) [2] achieve the top performance utilizing data augmentation, multi-task learning, and post-processing with an LSTM decoder. Chen et al. (2020) [3] proposes a pre-training stage for the encoder and combines the transformer decoder to achieve the second-best model. Perez-Castanos et al. (2020) [5] experiment resnet encoder and LSTM decoder with gammatone feature as an input. Pellegrini (2020) [4] propose Listen-Attend-Spell(LAS) architecture with listener encoder and speller decoder. Xu et al. (2020) [6] use a CRNN encoder and a GRU decoder with fine-tuning by reinforcement learning.

This year, the use of external data is allowed. Researchers can train the important sound-related feature representation using a massive amount of data such as AudioSet [8, 9]. Koizumi et al. (2020) [10] utilize a pre-trained VGGish model with a transformer decoder. Xu et al. (2021) [11] propose pre-trained CNN10 and CRNN5 as encoder networks with GRU decoder.

In this paper, we propose transfer learning followed by transformer architecture. With the transfer learning, our proposed model takes two pre-trained networks, 14-layers CNN (CNN14) and 54-layers ResNet (ResNet54), trained on AudioSet as the encoder part [12]. Those pre-trained networks achieve state-of-the-art performance on audio pattern recognition, and we expect our encoder network to compress important audio representation very well. Further, we train a transformer decoder using the Clotho v2.1 dataset for natural language generation. Finally, we experimented with three different audio feature preprocessing methods (log-mel spectrogram, CQT spectrogram, and gammatone filter spectrogram) in the ablation study.

2. PROPOSED METHOD

2.1. System Overview

Figure 1 describes the overview of our proposed system. The pre-trained CNN14 and ResNet54 are taken as an encoder using transfer learning [12]. We used a transformer decoder and trained our model using the Clotho v2.1 dataset. In the training stage, we experiment with three scenarios, 1) training all parameters from scratch without transfer learning, 2) fine-tuning the last block of the pre-trained network, and 3) training without fine-tuning the last block of the pre-trained network. After the training stages, we evaluate the performance of our system using a development set.

2.2. Pre-trained Audio Neural Networks using AudioSet

Kong et al. (2020) [12] propose Pre-trained Audio Neural Networks(PANNs) trained on the large-scale AudioSet dataset and make the 15 pre-trained models available to the public, including CNN14 and ResNet54. The AudioSet dataset includes over 5,000 hours of audio with 527 sound labels [8]. The audio clips from AudioSet data are extracted from YouTube videos. The training

*This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Ministry of Science and ICT (2020R1C1C1A01013020)

†This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Ministry of Science and ICT (2021R1F1A1056516)

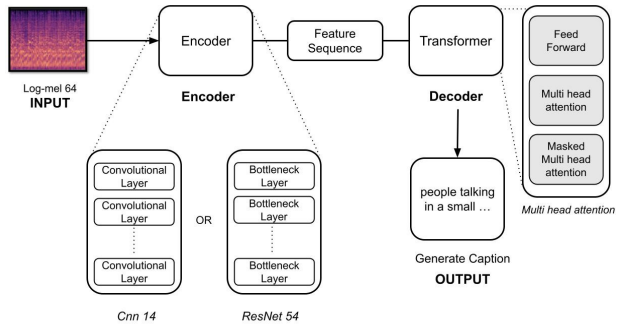

Figure 1: Model Architecture

dataset consists of 2,063,839 audio files, including a "balanced subset" of 22,160 audio files, where there are at least 50 audio files for each sound class. The evaluation dataset consists of 20,371 audio files. Audio files are padded to 10 seconds with silence if they are shorter than 10 seconds. The pre-trained models (CNN14 and REsNet54) are multi-label classification models for the 527 sound classes achieving state-of-the-art performance. These PANNs can be transferred to other audio-related tasks. We take CNN14 and ResNet54 as the encoder part for the AAC model. Table 1 and 2 describe CNN14 and ResNet54 model architecture, respectively.

2.3. Encoder-Decoder

Our proposed model also has the traditional encoder-decoder structure for AAC. Our model uses CNN14 or ResNet54 as an encoder and Transformer Decoder for natural language generation. The CNN14 and ResNet54 models were pre-trained networks (CNN14, ResNet54) learned from PANNs to AudioSet. We freeze the weights taken from the pre-trained networks and train the Transformer decoder with the Clotho data. Furthermore, we attempted to fine-tune the encoder network by unfreezing the last convolution block layers of CNN14 and ResNet54 to find the optimal model.

2.3.1. Encoder

Our model used CNN14 and Resnet54 as an encoder for feature extraction of input log-mel spectrogram [12]. CNN14 and Resnet54 models are pre-trained models from AudioSet, showing the highest mean average precision (mAP)[1] among the available pre-trained models. CNN14 and ResNet54 had the state-of-the-art mAP of 0.431 and 0.429, respectively [12]. Tables 1 and 2 show the structure of the CNN14 and ResNet54 that we used for ACC, respectively. The number after the "@" symbol indicates the number of feature maps. BottleneckB is an abbreviation for bottleneck block.

In Table 1, the 14-layer CNN consists of four convolution blocks, each having two 3 × 3 convolution layers with ReLU activation function and batch normalization, with an 2 × 2 average pooling layer between the blocks. The number of channels in the convolution blocks is 64, 128, 256, 512, 1024,2048, respectively.

[1] Average precision (AP) is defined as the area under the recall-precision curve of a specific class. The mean average precision (mAP) is the average value of AP over all classes.

Table 2 describes the ResNet54 architecture inspired by He et al. (2016) [13] for Audio tagging. Two convolutional layers and a downsampling layer are applied on the log-mel spectrogram to reduce the input log-mel spectrogram size. Additionally, We had three bottleneck blocks with 64 filters, four bottleneck blocks with 128 filters, 6 bottleneck blocks with 256 filters, and 3 bottleneck blocks with 512 filters. Finally, two 3 x 3 convolutions are applied. The BottleneckB in Table 2 is an abbreviation of bottleneck block.

Table 1: CNN14 architecture

CNN14
Log-mel spectrogram 64 mel bins
$(3 \times 3$ @64,BN,ReLU$)\times 2$
Pooling 2×2
$(3 \times 3$ @128,BN,ReLU$)\times 2$
Pooling 2×2
$(3 \times 3$ @256,BN,ReLU$)\times 2$
Pooling 2×2
$(3 \times 3$ @512,BN,ReLU$)\times 2$
Pooling 2×2
$(3 \times 3$ @1024,BN,ReLU$)\times 2$
Pooling 2×2
$(3 \times 3$ @2048,BN,ReLU$)*2$

Table 2: ResNet54 architecture

ResNet54
Log-mel spectrogram 64 mel bins
$(3 \times 3$ @512,BN,ReLU$)\times 2$
Pooling 2×2
(bottleneckB@64)$\times 3$
Pooling 2×2
(bottleneckB@128)$\times 4$
Pooling 2×2
(bottleneckB@256)$\times 6$
Pooling 2×2
(bottleneckB@512)$\times 3$
Pooling 2×2
$(3 \times 3$ @512,BN,ReLU$)\times 2$

2.3.2. Decoder

Figure 2 describes our transformer decoder architecture. It uses a standard transformer decoder consisting of multi-head self-attention as a decoder. The decoder uses a 2-layers transformer with a hidden dimension of 192 and 4 heads.

The input data of our decoder is the word embedding feature pre-trained using the word2vec model. The positional encoding is further applied. Next, it passes to the masked multi-head attention module and returns a query vector for the following multi-head attention module. The key and value vectors for the multi-head attention module are taken from the output of the encoder (CNN14 or ResNet54) network. After following the feed-forward network, we can get the output of the transformer block. The transformer block iterates two times, and then the output is fed into a dense layer and a softmax function to generate output probabilities of the caption words.

Table 3: Score for model performance on evaluation data

Model	$BLEU_1$	$BLEU_2$	$BLEU_3$	$BLEU_4$	$ROUGE_L$	METEOR	CIDEr	SPICE	**SPIDEr**
Baseline Model	0.378	0.119	0.050	0.017	0.263	0.078	0.075	0.028	0.051
CNN14 + Transformer (From Scratch)	0.466	0.262	0.156	0.092	0.309	0.137	0.208	0.087	0.148
ResNet54 + Transformer (From Scratch)	0.459	0.253	0.152	0.084	0.312	0.131	0.182	0.085	0.133
CNN14 + Transformer (Transfer-learning with fine-tuning)	0.552	0.364	0.244	0.159	0.378	0.168	0.395	0.118	0.257
ResNet54 + Transformer (Transfer-learning with fine-tuning)	0.546	0.358	0.239	0.156	0.373	0.166	0.379	0.113	0.246
CNN14 + Transformer (Transfer-learning)	0.564	0.376	0.254	0.163	0.388	0.177	0.441	0.128	0.285
ResNet54 + Transformer (Transfer-learning)	0.540	0.345	0.230	0.152	0.361	0.161	0.383	0.109	0.246

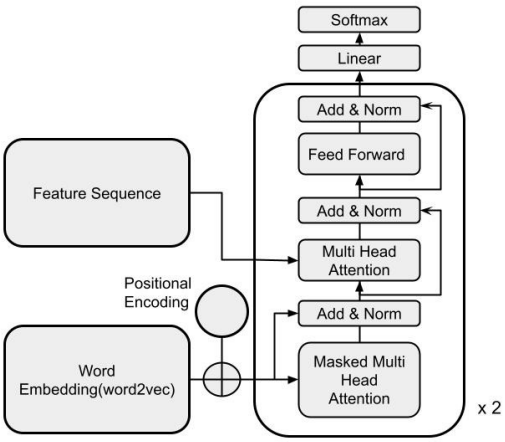

Figure 2: Transformer decoder

3. EXPERIMENTS

3.1. Dataset and Data Pre-processing

Clotho contains audio clips of CD quality (44.1 kHz sampling rate, 16-bit sample width) and five captions for each audio clip. The time duration of the audio clips ranges from 15 to 30 seconds, and the amount of words in each caption ranges from eight to 20 words. Clotho provides three splits for developing audio captioning methods, namely development, evaluation, and testing. The development and evaluation splits are freely available online, while the testing split is withheld for scientific challenges. In this work, we employ the development and evaluation splits of Clotho, having 2893 and 1045 audio clips, yielding 14465 and 5225 captions, respectively. We choose Clotho because it is built to offer audio content diversity, and extra care has been taken for eliminating spelling errors, named entities, and speech transcription in the captions. Additionally, Clotho is already employed at the DCASE 2020 audio captioning task6 [7].

The log-mel spectrogram feature is used for the input feature[14]. Audio data has 44.1kHz sampling frequency, and we apply a Hann window of 1024 size with 50% overlaps. From each window frame, we extract 64 log mel-band energies. We calculate the maximum time window number, T, among sample datasets for the number of time windows. We pad zero to the time dimension to have a fixed size T for the input feature on our model.

The word embedding is pre-trained using the Word2Vec model [15] via python package gensim [16]. Each caption sentence in the training set is used to form a training corpus.

Spec Augment [17] is applied as a data augmentation method for more robust training. With the Spec Augment, frequency masks and time masks are randomly applied onto the log-mel spectrogram before we feed the log-mel spectrogram input to the CNN14 or ResNet54 encoder.

3.2. Hyper-parameters and Training Procedure

3.2.1. Hyper-parameters

In training, a batch size of 8 is used with a learning rate of 10^{-4}. An l2 regularization is applied to all trainable parameters with factor λ = 10^{-6}. We use the Adam Optimizer [18] and apply the Stochastic Weight Averaging (SWA) method [19] to boost performance. Dropout in P = 0.2 is applied to ResNet54 encoder and Transformer decoder.

3.2.2. Training procedure

The training procedure consists of three parts. 1) Transfer learning step for encoder network, 2) training transformer decoder network while freezing the pre-trained encoder network, and 3) fine-tuning unfreezing last convolutional block parameters from encoder network.

We transfer two pre-trained networks, CNN14 and ResNet54, as our encoder network in the transfer learning stage. These two models are trained using log-mel spectrogram features using a large amount of AudoSet data. We freeze the encoder network for the following training stage.

In the training transformer decoder stage, we utilize the Clotho data. Each audio is combined with each one of five caption annotations and used as a sample. Each audio is used as one sample in the evaluation, and all five captions are used as a reference for metric computation. We used the 64 mel-band log-mel spectrogram of the audio as our input feature and then converted the amplitude into a decibel scale. A beam search with a beam size of 3 is implemented to achieve better decoding performance in the inference

Table 4: Ablation studies SPIDEr Score

Feature	Scratch	Transfer learning
Log-mel spectrogram	0.148	0.285
CQT spectrogram	0.15	-
gammatone spectrogram	0.2	-

stage. The Word2Vec model is trained with 1000 epochs, and the proposed model is trained using 30 epochs.

Finally, in the fine-tuning stage, we unfreeze the last convolution blocks of pre-trained networks, CNN14 and ResNet54. The fine-tuning is performed for 30 more epochs.

3.3. Evaluation Metrics

For the assessment of the performance of our method, we employ The proposed metrics from the audio captioning task at DCASE 2021 challenge. These metrics can be divided into two categories.

Firstly there are machine translation metrics, which are BLEU_n [20], ROUGE_L [21],and METEOR [22]. BLEU is a precision-based metric. It calculates a weighted geometric mean of a modified precision of n-grams between predicted and ground truth captions. Due to the calculation of the modified precision that favors short predicted captions, BLEU uses a penalty in the calculation of the geometric mean. This penalty penalizes predicted captions that are shorter than the ground truth. Typical lengths for n-grams are one to four, resulting in BLEU_n ($n \in \{1, 2, 3, 4\}$), respectively [20, 23]. ROUGE_L [21] is a Longest Common Subsequence (LCS) based metric. It calculates an F-measure using LCS between the predicted and ground truth caption. The F-measure is oriented towards recall using a value for the $\beta = 1.2$ in the F-measure calculation [21, 23]. METEOR [22] calculates a harmonic mean of precision and recall of segments of the captions between the predicted and ground truth captions. The recall is weighted significantly more than precision, and thus METEOR is considered a recall-based metric [23]. It employs alignment between the words of the predicted and ground truth captions and matches exact words, stems of words, synonyms, and paraphrases. The alignment is computed over segments of the captions between the ground truth and predicted captions while minimizing the number of chunks needed.

Then, the captioning metrics are CIDEr [24], SPICE [25], and a linear combination of these two metrics called SPIDEr [26]. CIDEr calculates a weighted sum of the cosine similarity between the predicted and ground truth captions for n-grams of length n with $\in [1, 4]$. The cosine similarity is calculated using Term Frequency Inverse Document Frequency (TF-IDF) weighting for each n-gram [24, 23]. SPIDEr is the average of CIDEr and SPICE, and it evaluates both fluency and semantic properties of the predicted captions.

3.4. Experimental Results

We experiment with three different scenarios, 1) training all parameters from scratch without transfer learning, 2) fine-tuning the last block of the pre-trained network, and 3) training without fine-tuning the last block of the pre-trained network. Table 3 shows the experimental results from the three different scenarios and the baseline. All scenarios of CNN14 Encoder+Transformer Decoder and ResNet54 Encoder+Transformer Decoder models have higher scores than the baseline model in all evaluation metrics. Also, All transfer learning scenarios have better performance than the training from the scratch scenario. This shows that the transferred encoders trained with sufficiently large amounts of audio data perform well on AAC. Between CNN14 and ResNet54 encoders, CNN14 encoder performed better than ResNet54 encoder in all evaluation metrics. We also experiment with fine-tuning the last convolution block of encoder networks (CNN14 and ResNet54 models). For CNN14 Encoder + Transformer Decoder, the transfer-learning model without fine-tuning works better than with the fine-tuning scenario in all evaluation metrics. On the other hand, for ResNet54 Encoder + Transformer Decoder, the model with fine-tuning has better performance with BLEU, ROUGE_L, and METEOR metrics which are machine translation metrics. However, for the captioning metrics such as SPICE, CIDEr, and SPIDEr scores, the result is inconclusive in that the models with fine-tuning have better SPICE scores but have worse CIDEr scores and have similar SPIDEr scores.

3.5. Ablation Studies

We have tested three feature extraction methods, 1) log-mel spectrogram, 2) constant Q transform (CQT) spectrogram, [27] and 3) Gammatone filter spectrogram. The Constant-Q-Transform (CQT) is a time-frequency representation where the frequency bins are geometrically spaced, and the Q-factors (ratios of the center frequencies to bandwidths) of all bins are equal. Gammatone filter spectrogram is computed by decomposing the input speech signal into the time-frequency (T-F) domain using a bank of Gammatone filters, followed by a down-sampling operation of the filter-bank responses along the time dimension [28].

We compare these three features without transfer learning because there are no pre-trained models with CQT and gammatone features. Table 4 shows the SPIDEr score performance using three different features. The model using CQT spectrogram (0.148) has similar SPIDEr performance with the model using log-mel spectrogram (0.15). However, the gammatone spectrogram (0.2) model performs better than the model using the log-mel spectrogram. Possibly, we may have better performance with the gammatone spectrogram feature if the pre-trained model using the gammatone spectrogram is available.

4. CONCLUSION

The DCASE community is hosting competitions for better AAC models in 2020 and 2021. In 2021, the use of external data is allowed. Thus, we propose transfer learning followed by a transformer approach. We adopt CNN14 and ResNet54 pre-trained on AudioSet data because it achieves state-of-the-art performance on audio pattern recognition. The pre-trained CNN14 or ResNet54 models are taken as encoder networks for informative audio feature extraction. With the transferred encoder and a transformer decoder, our proposed systems outperform the baseline system with all evaluation metrics. Further, we experiment with three training scenarios, 1) from scratch, 2) transfer learning, and 3) transfer learning with fine-tuning. Among them, the transfer learning of CNN14 encoder without fine-tuning works the best, achieving a SPIDEr score of 0.285.

5. REFERENCES

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

[2] D. Takeuchi, Y. Koizumi, Y. Ohishi, N. Harada, and K. Kashino, "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)*, 2020, pp. 190–194.

[3] K. Chen, Y. Wu, Z. Wang, X. Zhang, F. Nian, S. Li, and X. Shao, "Audio captioning based on transformer and pre-trained cnn," in *Proceedings of the Detection and Classification of Acoustic Scenes and Events 2020 Workshop (DCASE2020)*, 2020, pp. 21–25.

[4] T. Pellegrini, "IRIT-UPS DCASE 2020 audio captioning system," DCASE2020 Challenge, Tech. Rep., June 2020.

[5] S. Perez-Castanos, J. Naranjo-Alcazar, P. Zuccarello, and M. Cobos, "Listen carefully and tell: An audio captioning system based on residual learning and gammatone audio representation," in *Proceedings of the Detection and Classification of Acoustic Scenes and Events 2020 Workshop (DCASE2020)*, 2020, pp. 150–154.

[6] X. Xu, H. Dinkel, M. Wu, and K. Yu, "A crnn-gru based reinforcement learning approach to audio captioning," in *Proceedings of the Detection and Classification of Acoustic Scenes and Events Workshop (DCASE2020)*, 2020, pp. 225–229.

[7] K. Drossos, S. Lipping, and T. Virtanen, "Clotho: An audio captioning dataset," in *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2020, pp. 736–740.

[8] J. F. Gemmeke, D. P. Ellis, D. Freedman, A. Jansen, W. Lawrence, R. C. Moore, M. Plakal, and M. Ritter, "Audio set: An ontology and human-labeled dataset for audio events," in *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2017, pp. 776–780.

[9] Q. Kong, C. Yu, Y. Xu, T. Iqbal, W. Wang, and M. D. Plumbley, "Weakly labelled audioset tagging with attention neural networks," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 27, no. 11, pp. 1791–1802, 2019.

[10] Y. Koizumi, R. Masumura, K. Nishida, M. Yasuda, and S. Saito, "A transformer-based audio captioning model with keyword estimation," *arXiv preprint arXiv:2007.00222*, 2020.

[11] X. Xu, H. Dinkel, M. Wu, Z. Xie, and K. Yu, "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)*. IEEE, 2021, pp. 905–909.

[12] Q. Kong, Y. Cao, T. Iqbal, Y. Wang, W. Wang, and M. D. Plumbley, "Panns: Large-scale pretrained audio neural networks for audio pattern recognition," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 28, pp. 2880–2894, 2020.

[13] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[14] B. Thornton, "Audio recognition using mel spectrograms and convolution neural networks," 2019.

[15] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," *arXiv preprint arXiv:1301.3781*, 2013.

[16] R. Rehurek and P. Sojka, "Software framework for topic modelling with large corpora," in *In Proceedings of the LREC 2010 workshop on new challenges for NLP frameworks*. Citeseer, 2010.

[17] D. S. Park, W. Chan, Y. Zhang, C.-C. Chiu, B. Zoph, E. D. Cubuk, and Q. V. Le, "Specaugment: A simple data augmentation method for automatic speech recognition," *arXiv preprint arXiv:1904.08779*, 2019.

[18] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[19] P. Izmailov, D. Podoprikhin, T. Garipov, D. Vetrov, and A. G. Wilson, "Averaging weights leads to wider optima and better generalization," *arXiv preprint arXiv:1803.05407*, 2018.

[20] K. Papineni, S. Roukos, T. Ward, and W.-J. Zhu, "Bleu: a method for automatic evaluation of machine translation," in *Proceedings of the 40th annual meeting of the Association for Computational Linguistics*, 2002, pp. 311–318.

[21] C.-Y. Lin, "Rouge: A package for automatic evaluation of summaries," in *Text summarization branches out*, 2004, pp. 74–81.

[22] A. Lavie and A. Agarwal, "Meteor: An automatic metric for mt evaluation with high levels of correlation with human judgments," in *Proceedings of the second workshop on statistical machine translation*, 2007, pp. 228–231.

[23] X. Chen, H. Fang, T.-Y. Lin, R. Vedantam, S. Gupta, P. Dollár, and C. L. Zitnick, "Microsoft coco captions: Data collection and evaluation server," *arXiv preprint arXiv:1504.00325*, 2015.

[24] R. Vedantam, C. Lawrence Zitnick, and D. Parikh, "Cider: Consensus-based image description evaluation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 4566–4575.

[25] P. Anderson, B. Fernando, M. Johnson, and S. Gould, "Spice: Semantic propositional image caption evaluation," in *European conference on computer vision*. Springer, 2016, pp. 382–398.

[26] S. Liu, Z. Zhu, N. Ye, S. Guadarrama, and K. Murphy, "Improved image captioning via policy gradient optimization of spider," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 873–881.

[27] T. Lidy and A. Schindler, "Cqt-based convolutional neural networks for audio scene classification," in *Proceedings of the detection and classification of acoustic scenes and events 2016 workshop (DCASE2016)*, vol. 90, 201 organization=IEEE Budapest, pp. 1032–1048.

[28] B. Ayoub, K. Jamal, and Z. Arsalane, "Gammatone frequency cepstral coefficients for speaker identification over voip networks," in *2016 International Conference on Information Technology for Organizations Development (IT4OD)*. IEEE, 2016, pp. 1–5.