TEXT-TO-AUDIO RETRIEVAL VIA LARGE-SCALE CONTRASTIVE TRAINING

Yusong Wu[1,2*], *Tianyu Zhang*[1,2,*], *Ke Chen*[3,*],

[1] Université de Montréal
[2] Mila
{wu.yusong, tianyu.zhang}@mila.quebec
[3] University of California San Diego,
knutchen@ucsd.edu

ABSTRACT

Although there is an abundance of data available on the internet, audio data is still limited in terms of dataset size and label precision. Scaling the size of audio datasets would therefore be one of the most valuable ways to develop models for better audio understanding. In this report, we propose a pipeline to better learn the audio understanding mechanism by combining audio data with more abundantly available natural language descriptions. We collected a mixed dataset consisting of over 2 million data pairs and trained a contrastive model based on Contrastive Language–Image Pre-training (CLIP) in order to discover correspondence between audio and text. As an audio encoder, we use HTS-AT as a transformer-based model and PANN and a CNN-based model, and as a text encoder, we employ the frozen pre-trained CLIP text encoder. The resulting models are submitted to Task 6B of the DCASE 2022 challenge and achieve a mAP@10 score of at least 0.214.

Index Terms— Contrastive Learning, Audio Understanding, Text-to-audio Retrieval

1. INTRODUCTION

Data is the lifeblood of modern deep learning models. However, audio data is haunted by the limited amount of training data, which is the result of costly data collection and labor-intensive data labeling. Despite this, audio data is one of the most prevalent data in the internet. Thus, training on audio available on the internet without the need of any supervision would be of great help to learn a good audio understanding model.

A successful solution for training on "data in the wild" is multi-modal contrastive training, symbolized by the recently proposed Contrastive Language–Image Pre-training (CLIP) [1]. CLIP learns the correspondence between text and image by projecting them into a shared latent space and employing contrastive learning by treating the ground-truth image-text pair as the positive sample and all others as negative samples. Unlike training on the single modal data in an unsupervised manner [2, 3, 4], the training paradigm of CLIP is not limited by training labels (class label, caption or filename) and shown its strong capability in robustness. Additionally, multi-modal training in CLIP also shows great success in downstream tasks such as information retrieval [5], text-guided generation [6, 7, 8] or captioning [9] that are benefited most from learning with natural language supervision.

In this report, we follow the insights of CLIP and propose to train models on a large mixed dataset of audio-text pairs to combat data scarcity. We collect more than 2 million data pairs consisting of 6182 hours of audios from publicly available datasets, websites and sound effect libraries. To train models on such mixed dataset, we refer to the architecture from CLIP, adapt it as a **Contrastive Language–Audio Pre-training (CLAP)** model. There are existing works on audio-text contrastive learning or expanding CLIP to audio domain [10, 11, 12, 13]. On the basis of similar insights, the proposed model is trained with natural language descriptions as text supervision other than limited in class labels.

We adopt a CNN-based model — PANN [14], and a transformer-based model — HTS-AT [15], as our choices of audio encoder to encode the audio data, while loading and freezing the pre-trained text encoder from CLIP to encode the text data. The whole CLAP model is used for DCASE 2022 Task 6B "Language-Based Audio Retrieval". We ensemble models of different settings to construct 4 systems for submission, and report text-to-audio retrieval performance for each submission system on Clotho dataset [16], resulting in the best mAP@10 score of 0.214. The code for the proposed system is open-source online[1].

2. DATASET AND DATA LOADING

2.1. Dataset

We collect a mixed dataset to train the CLAP model. The mixed dataset consists of more than 2 million audio tracks in a total duration of 6182 hours and their corresponding texts. The detail of these data is listed in Table 1. For Clotho, AudioCaps and AudioSet, we use their designated validation and test sets. For other data sources, we randomly split 90% of data as training set (shown in Table 1), and the left 10% as test set.

2.2. Data Pre-process

The goal of the data pre-processing is to convert audios and labels from different datasets into a unified and generic format stored on the file system for loading by various models.

For audio files in each dataset, we convert them into Free Lossless Audio Codec (FLAC) format with 48kHz sample rate, in order to reduce the information loss and save the storage space (cf. wav. format).

For labels, tags, or captions, we store them into json files. For datasets with only labels (e.g., AudioSet), we prompt them into text by placing the label as "Sound of label_1, label_2, and label_3, ..., and so on".

*equal contribution

[1] https://github.com/LAION-AI/CLAP

Work	Dataset	# of Samples	Duration (hours)	Data Used
Lee et al. [10]	VGGSound [17]	200,000	555.56	Labels, audio
Wav2CLIP [11]	VGGSound [17]	200,000	555.56	Video frames, audio
Audioclip [12]	AudioSet [18]	1,912,134	5311.48	Labels, audio
Ours	BBC Sound Effects [19]	14,376	430.89	1 caption per audio, audio
	Clotho [16]	3,839	23.99	5 captions per audio, audio
	AudioCaps [20]	46,074	126.20	1 caption per audio, audio
	AudioSet [18]	1,912,134	5311.48	Labels, audio
	Free To Use Sounds [21]	5,733	158.15	Filename, audio
	We Sound Effects [22]	439	10.24	Filename, audio
	Paramount Motion Sound Effects [23]	4,439	17.45	Filename, audio
	Sonniss Game Audio Effects [24]	5,005	76.91	Filename, audio

Table 1: The dataset used to train CLAP comparing to previous works.

2.3. Data Loading

The input to the model is log-mel-spectrogram (audio), and word tokens (text). All the audio in 10 seconds with a sample rate of 48kHz, resulting in the sample length of 480,000. We adopt the zero-padding for the audio shorter than 10 seconds and randomly select 10 seconds piece for the audio longer than 10 seconds. When converting audio to mel-spectrogram, we use 480 frame size and 64 frequency bins, resulting in a 2D matrix of shape $(T = 1000, F = 64)$. For audio samples having more than one text captions, we randomly select one of the caption each time when loading samples from data. We augment the data by applying the frequency masking on 8 bins and time masking on 128 frames.

3. MODEL

3.1. General Architecture

Figure 1 depicts the general architecture of our proposed Contrastive Language-Audio Pre-training (CLAP) framework. Compared to the original CLIP architecture, we make two modifications to adapt it to the contrastive framework between text and audio data.

The original image encoder is replaced with the audio encoder, which will be introduced in the following section. The audio encoder can be selected from PANN [14], HTS-AT [15], and, if given, other audio classification models. For the text encoder, we employ the same model as demonstrated in OpenAI's `Vit-B-16`[2] model checkpoint. It is a transformer encoder with 12 layers, 8 heads, 77 context length, 512-dimensional hidden state, and 49408 words in its vocabulary.

Second, to prevent interfering with the prior and to make the model more robust, we freeze the text encoder's parameters, as OpenAI has already trained it on a significantly bigger text data set. In the meantime, this freezing paradigm can also help us achieve semantic latent space homology between learned audio and text embeddings.

3.2. Audio Encoder

For the choice of audio embeddings that used to perform the contrastive learning with text embeddings, we seek an embedding to model and characterize the acoustic features and semantic content of audio. Audio classification is a typical task that refer to classify sound event labels of given audio samples. Therefore, we employ two audio classification models in our CLAP framework: (1) Pretrained Audio Neural Networks (PANN) [14], as a CNN-based audio classifier, and (2) Hierarchical Token-Semantic Audio Transformer (HTS-AT) [15], as a transformer-based classifier.

3.2.1. PANN

As shown in the left of Figure 2, PANN contains VGG-like CNNs [25] to convert an audio mel-spectrogram into a (T, C) featuremap, where T is the number of time frames and C is the number of sound event classes. The structures of CNN blocks can be varied from different numbers (e.g, CNN-6, CNN-10, CNN-14, etc.). In this report, we use the CNN-14 structure, which contains 7 downsampling CNN blocks and 7 upsampling CNN blocks to process the audio spectrogram inputs. The model averages the featuremap over the time axis to obtain a final probability vector $(1, C)$ and computes the binary cross-entropy loss between it and the groudtruth label. Since CNNs can capture the information in each time window, the featuremap (T, C) is empirically regarded as a presence probability map of each sound event at each time frame. When determining the latent source embedding for the following pipeline, the penultimate layer's output (T, L) can be used to obtain its averaged vector $(1, L)$ as the audio embedding. In this report, we set $L = 2048$ and C is not used in the training stage because we do not explicitly train the audio encoder in the audio classification scenario (but in the contrastive learning scenario with the text encoder).

3.2.2. HTSAT

HTS-AT is a hierarchical token-semantic transformer for audio classification. The token-semantic module [26] have been widely used in the image classification and segmentation task and achieve better performance. HTS-AT applies swin-transformer [27] and token-semantic module into the audio classification task. In the right of Figure 2, a mel-spectrogram is cut into different patch tokens with a patch-embed CNN and sent into the transformer in order. The time and frequency lengths of the patch is equal as $P \times P$.

Further, to better capture the relationship between frequency bins of the same time frame, HTS-AT first splits the mel-spectrogram into windows $w_1, w_2, ..., w_n$ and then splits the

[2] `https://openaipublic.azureedge.net/clip/models/ViT-B-16.pt`

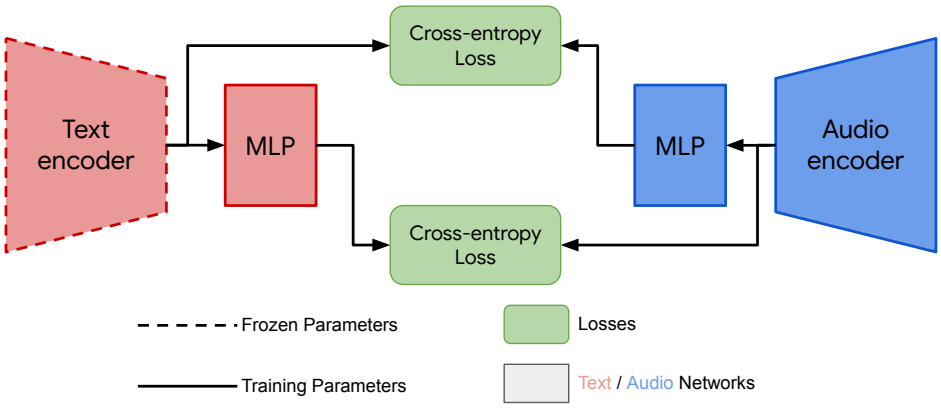

Figure 1: The architecture of CLAP, to obtain the embedding from audio encoder and text encoder via contrastive learning.

patches in each window. The order of tokens Q follows **time→frequency→window** as:

$$Q = \{q^{w_1}_{1,1}, q^{w_1}_{1,2}, ..., q^{w_1}_{1,t}, q^{w_1}_{2,1}, q^{w_1}_{2,2}, ..., q^{w_1}_{2,t}, ..., q^{w_1}_{f,t}, \quad (1)$$

$$q^{w_2}_{1,1}, q^{w_2}_{1,2}, ..., q^{w_2}_{1,t}, q^{w_2}_{2,1}, q^{w_2}_{2,2}, ..., q^{w_2}_{2,t}, ..., q^{w_2}_{f,t}, \quad (2)$$

$$q^{w_3}_{1,1}, ..., q^{w_3}_{f,t}, q^{w_4}_{1,1}, ..., q^{w_4}_{f,t}, ..., q^{w_n}_{f,t}\} \quad (3)$$

Where $t = \frac{T}{P}, f = \frac{F}{P}$, n is the number of time windows, and $q^{w_k}_{i,j}$ denotes the patch in the position shown by Figure 2.

The patch tokens pass through several network groups, each of which contains several transformer-encoder blocks. Between every two groups, a patch-merge layer is applied to reduce the number of tokens to construct a hierarchical representation. Each transformer-encoder block is a swin-transformer block with the shifted window attention module [27], a modified self-attention module to improve the training efficiency. As illustrated in Figure 2, the shape of the patch tokens is reduced by 8 times from $(\frac{T}{P} \times \frac{F}{P}, D)$ to $(\frac{T}{8P} \times \frac{F}{8P}, 8D)$ after 4 network groups.

Then, HTS-AT applies a token-semantic 2D-CNN [26] to further process the reshaped output $(\frac{T}{8P}, \frac{F}{8P}, 8D)$ for the audio classification target. The audio embedding, at the same time, is produced by averaging the reshaped output into a L-dimension vector with an average-pooling layer. The structure of the swin transformer can be varied from different capability (e.g., tiny, base, large, etc.). In this report, we follow the tiny setting to set $D = 96$, $L = 768$, $P = 4$, and the number of transformer blocks in 4 groups is $2, 2, 6, 2$.

3.3. Training Paradigm

For training objectives, we add two MLP layers after the audio embeddings $Y_a = \{y_{a1}, y_{a2}, ..., y_{am}\}$ and the text embeddings Y_t as Y^m_a and Y^m_b. According to [11], this would boost the converging speed and we introduce a new clip loss function by computing the original clip loss between different combinations of output embeddings:

$$L = \frac{f(Y_a, Y^m_t, o_1) + f(Y^m_a, Y_t, o_2)}{2} \quad (4)$$

$$f(a, b, o)_{local} = \frac{1}{2N} \sum_{i=1}^{N} CE(o \cdot a_i \times b.T, l) + CE(o \cdot b_i \times a.T, l) \quad (5)$$

$$f(a, b, o)_{global} = CE(o \cdot a \times b.T, l) \quad (6)$$

Where l is the label (i.e. the correct index of the audio/text embeddings, in here l is a diagonal matrix), o_1 and o_2 are learnable logit parameters for scaling the cross entropy loss. $CE()$ is the cross entropy loss function. f is the function to compute the original CLIP loss, which involves two formats: (1) The global format is to directly compute the CLIP loss, and (2) The local format is to deal with the multi-GPU training, where the total batch embeddings Y_a, Y_t, Y^m_a and Y^m_t are distributed into different sub-batches on different GPUs (e.g., $Y_{a1}, Y_{a2}, ..., Y_{aN}$, note that this is different from the above $y_{a1}, y_{a2}, ..., y_{am}$). Then, the local loss could compute the CLIP loss between each sub-batches' embeddings with the total numbers of embeddings.

4. SUBMISSION

4.1. Metrics

We evaluate performance of the model according to retrieval-based metrics. Specifically, we use mean rank, median rank, R@1, R@5, R@10, and mAP@10 for both text-to-audio and audio-to-text retrieval as the metrics we monitor during training. Mean rank defines the average rank of the ground truth for a specific query. Median rank defines the median of the rank of the ground truth for a specific query. R@k calculates the recall score among the top-k retrieved results, averaged across all queries where k can be 1, 5 and 10. mAP@10 computes the average precision among the top-10 retrieved results defined by the query, averaged across all caption queries. For each run, we save 3 models with best text-to-audio mAP@10 performance on Clotho evaluation set.

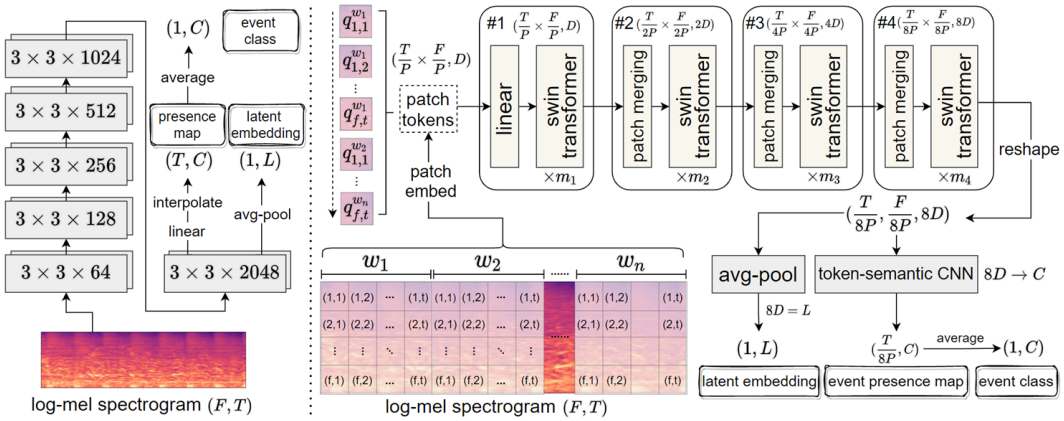

Figure 2: The network architecture of audio classification models. Left: PANN [14]; Right: HTS-AT [15]. All CNNs are named as [2D-kernel size × channel size].

Model	Mean Rank	Median Rank	R@1	R@5	R@10	**mAP@10**
Baseline	-	-	0.03	0.11	0.19	0.07
System 1	50.670	14.000	0.126	0.334	0.452	0.214
System 2	48.665	14.000	0.124	0.326	0.451	0.211
System 3	48.315	14.000	0.124	0.331	0.451	0.212
System 4	49.171	13.000	0.124	0.327	0.455	0.212

Table 2: The systems for submission and their text-to-audio retrieval performance on Clotho evaluation set.

4.2. Training Hyperparameters

For the training hyperparameters, we trained all models with the AdamW [28] optimizer ($\beta_1 = 0.9$, $\beta_2 = 0.999$, $\epsilon = 1e-8$), the learning rate $1e-3$, and the cosine annealing learning scheduler with the warm-up step 10000. We implemented our proposed model in PyTorch[3] and trained it on 8 NVIDIA Telsa A-100 GPU. The batch size is 1472 (184×8). All models are converaged in 400 epochs.

4.3. Submission Systems

We train 4 CLAP models with HTS-AT audio encoder by 4 different random seeds, and 2 CLAP models with PANN audio encoder. In our submission, we list 4 systems by ensembling these trained models with different settings, and report their text-to-audio retrieval performance in Table 2:

- System 1: the ensemble model of all HTS-AT CLAPs and PANN CLAPs;
- System 2: the ensemble model of all HTS-AT CLAPs;
- System 3: the ensemble model of all HTS-AT CLAPs and one top-performance CNN CLAPs;
- System 4: the ensemble model of two top-performance HTS-AT CLAPs and one top-performance CNN CLAPs.

For each model, we ensemble the first three checkpoints with top text-to-audio mAP@10 score. The reason to choose these combinations is to explore if CNN CLAPs and HTS-AT CLAPs learn different understandings of audio data and text data and consider the collaboration of two architectures.

[3] https://pytorch.org/

5. ACKNOWLEDGMENT

We would like to thank professor Irina Rish, Christoph Schuhmann, Shlomo Dubnov, Taylor Berg-Kirkpatrick, Romain Beaumont for their support of this project. We would like to thank Richard Vencu for his support of dataset collections. We would also like to thank the support of computational resources from LAION[4] and stability.ai[5]. This project would not have been possible without the contribution of the following open source projects: Pytorch[6], Open-clip[7], and Numpy[8].

6. REFERENCES

[1] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark, *et al.*, "Learning transferable visual models from natural language supervision," in *International Conference on Machine Learning*. PMLR, 2021, pp. 8748–8763.

[2] K. He, X. Chen, S. Xie, Y. Li, P. Dollár, and R. Girshick, "Masked autoencoders are scalable vision learners," 2021.

[4] https://laion.ai/
[5] https://stability.ai/
[6] https://pytorch.org/
[7] https://github.com/mlfoundations/open_clip
[8] https://github.com/numpy/numpy

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

[4] A. Baevski, W.-N. Hsu, Q. Xu, A. Babu, J. Gu, and M. Auli, "data2vec: A general framework for self-supervised learning in speech, vision and language," 2022.

[5] "clip-retrieval," https://github.com/rom1504/clip-retrieval, 2022, [Online; accessed 01-May-2022].

[6] A. Nichol, P. Dhariwal, A. Ramesh, P. Shyam, P. Mishkin, B. McGrew, I. Sutskever, and M. Chen, "Glide: Towards photorealistic image generation and editing with text-guided diffusion models," *arXiv preprint arXiv:2112.10741*, 2021.

[7] A. Ramesh, M. Pavlov, G. Goh, S. Gray, C. Voss, A. Radford, M. Chen, and I. Sutskever, "Zero-shot text-to-image generation," in *International Conference on Machine Learning*. PMLR, 2021, pp. 8821–8831.

[8] A. Ramesh, P. Dhariwal, A. Nichol, C. Chu, and M. Chen, "Hierarchical text-conditional image generation with clip latents," *arXiv preprint arXiv:2204.06125*, 2022.

[9] R. Mokady, A. Hertz, and A. H. Bermano, "Clipcap: Clip prefix for image captioning," *arXiv preprint arXiv:2111.09734*, 2021.

[10] S. H. Lee, N. Lee, C. Kim, W. Ryoo, J. Kim, S. H. Yoon, and S. Kim, "Audio-guided image manipulation for artistic paintings," in *Workshop on Machine Learning for Creativity and Design at 35th Conference on Neural Information Processing Systems (NeurIPS 2021)*, 2021.

[11] H.-H. Wu, P. Seetharaman, K. Kumar, and J. P. Bello, "Wav2clip: Learning robust audio representations from clip," in *ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2022, pp. 4563–4567.

[12] A. Guzhov, F. Raue, J. Hees, and A. Dengel, "Audioclip: Extending clip to image, text and audio," *arXiv preprint arXiv:2106.13043*, 2021.

[13] D. C.-E. Lin, A. Germanidis, C. Valenzuela, Y. Shi, and N. Martelaro, "Soundify: Matching sound effects to video," *arXiv preprint arXiv:2112.09726*, 2021.

[14] 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.

[15] K. Chen, X. Du, B. Zhu, Z. Ma, T. Berg-Kirkpatrick, and S. Dubnov, "HTS-AT: A hierarchical token-semantic audio transformer for sound classification and detection," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2022.

[16] 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.

[17] H. Chen, W. Xie, A. Vedaldi, and A. Zisserman, "Vggsound: A large-scale audio-visual dataset," in *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2020, pp. 721–725.

[18] 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.

[19] "BBC Sound Effects," https://sound-effects.bbcrewind.co.uk/, 2022, [Online; accessed 01-May-2022].

[20] C. D. Kim, B. Kim, H. Lee, and G. Kim, "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, Volume 1 (Long and Short Papers)*, 2019, pp. 119–132.

[21] "Free To Use Sounds," https://www.freetousesounds.com/product/all-in-one-sound-library-bundle/, 2022, [Online; accessed 01-May-2022].

[22] "We Sound Effects," https://wesoundeffects.com/we-sound-effects-bundle-2020/, 2022, [Online; accessed 01-May-2022].

[23] "Paramount Motion Sound Effects," https://www.paramountmotion.com/odeon-sound-effects, 2022, [Online; accessed 01-May-2022].

[24] "Sonniss Game Audio Effects," https://sonniss.com/gameaudiogdc, 2022, [Online; accessed 01-May-2022].

[25] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *3rd International Conference on Learning Representations, ICLR 2015*, 2015.

[26] W. Gao, F. Wan, X. Pan, Z. Peng, Q. Tian, Z. Han, B. Zhou, and Q. Ye, "Ts-cam: Token semantic coupled attention map for weakly supervised object localization," in *Premier International Computer Vision Event, ICCV 2021*, 2021.

[27] Z. Liu, Y. Lin, Y. Cao, H. Hu, Y. Wei, Z. Zhang, S. Lin, and B. Guo, "Swin transformer: Hierarchical vision transformer using shifted windows," 2021.

[28] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *3rd International Conference on Learning Representations, ICLR 2015*.