CLIP2CLAP: Joint Embedding Space Alignment for Image to Audio Generation (In Progress) *

Yau-Meng Wong
Columbia University
New York City
yw3809@columbia.edu

Abstract

Training models on large files such as image and audio is a computationally expensive task. Faced with this dilemma, one might turn to a pretrained model that can already solve the task at hand. This method only works if such a model exists. For an uncommon task like image-conditioned audio generation, this likely won't be the case. Instead, you may have to use multiple models in sequence. Embedding space alignment is one method to tie pretrained models together. With multimodal latent spaces, such as OpenAI's CLIP, aligning two latent spaces along one mode, such as text, has the side effect of aligning every other mode pair between the two spaces. This has been proven to be effective by Meta's ImageBind[1]. In their paper they showed that a joint embedding space over multiple modes (images, text, audio, depth, thermal, and IMU data) could be trained on data of each mode paired solely with images. In this case I'll be using text as the "binding" data between images and audio. I'll be taking two pretrained models, OpenAI's, an image/text joint encoder, and LAION's, an audio/text joint encoder, and aligning them using only text data CLIP[2]CLAP[3]. Then, with pretrained audio generation model AudioLDM that takes CLAP audio embeddings as conditioning, I will be able to do image-conditioned audio generation[4].

Keywords Embedding Alignment · Image to Audio

1 Introduction

The main task is to align the CLIP and CLAP embedding spaces. At its core it's basically a machine translation problem to and from the same language, for a single embedding. I've tried various combinations of different model architectures, objective functions, and datasets. The current iteration is just what I'm currently training. It may not be the best model so far, but I have yet to find a good evaluation metric other than how true the generated audio is to the image prompt, to my own ear. So far none of the models have been any good.

2 Methodology

2.1 Previous Work

The current model is inspired by three main works. The model itself is a simple 4 layer feed forward network. I also tried a Transformer encoder, and I'll likely try it again, but I decided to train a linear model because of the finding in "Word Translation Without Parallel Data" that embedding spaces of similar languages could be aligned using a single linear layer[5]. This is shown in Figure 1, which comes directly from the paper. As it seemed to work with cross-lingual tasks, I thought it was reasonable to assume a linear model could translate between the same language. The current objective function is inspired by CLIP's own objective function, introduced in "Learning Transferable Visual Models From Natural Language Supervision". The task of creating a joint embedding space is similar to that of aligning embedding spaces in that the result should map similar pairs close together (by whatever distance metric
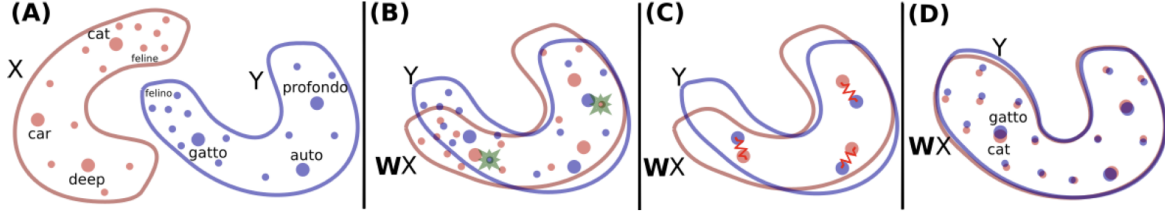

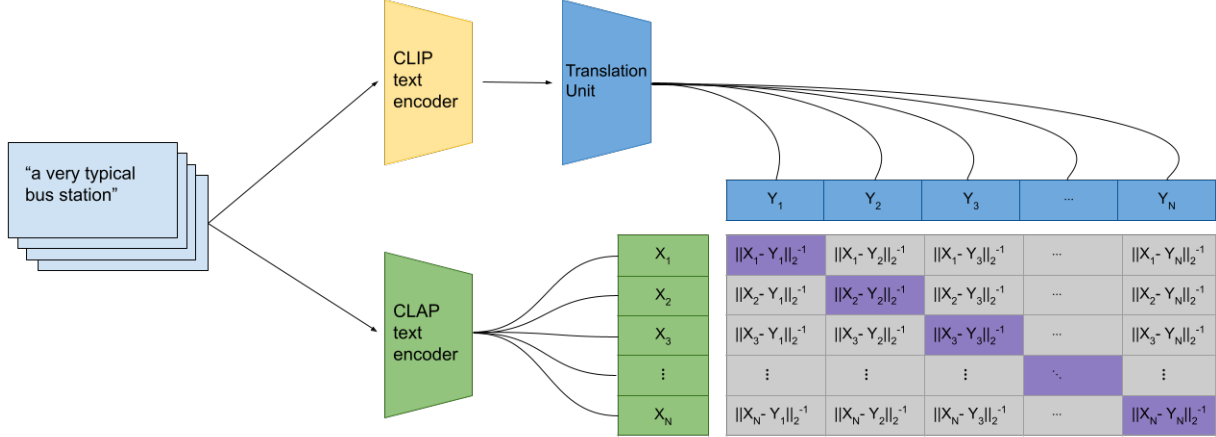Figure 1: Alignment visualization of English and Spanish latent spaces.

Figure 2: Training architecture.

is chosen), and dissimilar pairs farther. In their case they used cosine similarity, but I chose to use inverse euclidian distance. I also experimented with the concept of margin introduced in "Dimensionality Reduction by Learning an Invariant Mapping", though not in the way it was demonstrated in the paper [6]. Whether it made a positive impact on model performance is unclear.

2.2 Data

I first tried training on 60000 image captions from Common Objects in Context (COCO), but realized this was ideal for multiple reasons[7]. The data set was neither large nor varied enough. It was paired with a set of images, each image having 5 different captions. This was dangerous when using a contrastive loss function over randomly shuffled data, as ideally all the samples in a batch are somewhat distinct. I switched to using Google's Conceptual Captions (GCC) data set, which contains 3 million unique descriptive captions[8]. I chose GCC because it was larger, more varied, and contained a larger vocabulary, meaning the model would be exposed to a more diverse selection of words.

2.3 Model Architecture

The translation unit is just four feed forward layers with Tanh activation functions. The logits are clamped between +-0.2, as I found the distribution of values in the CLAP embeddings rarely exceeded this. At train time each batch of 32 image captions are fed into both the CLIP and CLAP text encoders, then the embedding generated by CLIP is fed through the translation unit. The inverse euclidean distance is calculated for each pair of translated CLIP and CLAP embeddings, stored in a matrix as shown in Figure 2, then cross entropy loss is performed along every column and every row, with the target label being the row number or column number. The loss is summed and then back propagation is performed, updating only the parameters of the translator unit. At inference time, an image is passed to the CLIP image encoder, then the resulting embedding is passed through the translator unit, then the translated embedding is passed as a prompt to the audio generation model, in this case AudioLDM.
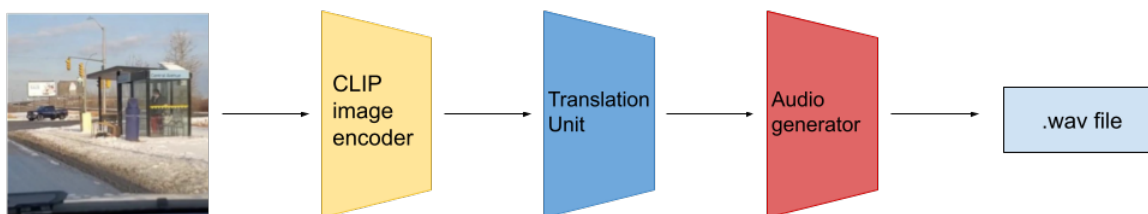
Figure 3: Inference architecture.

3   Results

This is still a work in progress, and the results thus leave much to be desired. The process is slowed by the fact that each epoch (one pass through the GCC data set) take about 12 hours. Here are a set of images and the resulting generated audio. Info regarding specific pretrained checkpoints/model parameters are in the code on Github. Once I come up with a better model I'll add better documentation and create a Colab notebook for users to test the model out. Here are a few images and the resulting audio generated by various models I trained, all with a similar structure as the one described above. These are hand selected, but I'm including the best and worst examples. For reference, I prompted the audio generation model with just a normal text prompt describing the image. This acts as a control.

The code for may be found at

https://github.com/y4umeng/CLIP2CLAP

Acknowledgments

References

[1] Rohit Girdhar, Alaaeldin El-Nouby, Zhuang Liu, Mannat Singh, Kalyan Vasudev Alwala, Armand Joulin, and Ishan Misra. Imagebind: One embedding space to bind them all. 2023.

[2] Alec Radford, Jong Wook Kim, and Chris Hallacy et al. Learning transferable visual models from natural language supervision. 2021.

[3] Yusong Wu, Ke Chen, and Tianyu Zhang et al. Large-scale contrastive language-audio pretraining with feature fusion and keyword-to-caption augmentation. 2023.

[4] Haohe Liu, Zehua Chen, Yi Yuan, Xinhao Mei, Xubo Liu, Danilo Mandic, Wenwu Wang, and Mark D. Plumbley. Audioldm: Text-to-audio generation with latent diffusion models. 2023.

[5] Alexis Conneau, Guillaume Lample, Marc'Aurelio Ranzato, Ludovic Denoyer, and Hervé Jégou. Word translation without parallel data. 2018.

[6] R. Hadsell, S. Chopra, and Y. LeCun. Dimensionality reduction by learning an invariant mapping. In *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, volume 2, pages 1735–1742, 2006.

[7] Tsung-Yi Lin, Michael Maire, Serge Belongie, Lubomir Bourdev, Ross Girshick, James Hays, Pietro Perona, Deva Ramanan, C. Lawrence Zitnick, and Piotr Dollár. Microsoft coco: Common objects in context. 2015.

[8] Piyush Sharma, Nan Ding, Sebastian Goodman, and Radu Soricut. Conceptual captions: A cleaned, hypernymed, image alt-text dataset for automatic image captioning. In *Proceedings of ACL*, 2018.

CLIP2CLAP

Table 1: Results (WIP)

Image	Audio	Control	Translation Unit Architecture
	Soundcloud link	"Two seagulls on a pier"	Two layer feed forward trained on COCO
	Soundcloud link	"Hatsune Miku on a white background"	Two layer feed forward trained on COCO
	Soundcloud link	"A woman with a fancy hat on"	Four layer feed forward trained on GCC
	Soundcloud link	"A cute orange cat"	Four layer feed forward trained on GCC
	Soundcloud link	A hammer with a white background	Transformer encoder trained on COCO.