ECE 549 — Final Project Final Report

Survey of Audio-Visual Representation Learning: Audio-Visual Correspondence Task

Kai Chieh (Jeff) Chang
Universiry of Illinois at Urbana Champaign
Electrical and Computer Engineering
kcchang3@illinois.edu

Abstract

This project is a survey on a particular method to extract audio and visual information using self-supervised learning: "Audio-Visual Correspondence" (AVC) task. I built the underlying audio-visual neural network to train on unlabeled image and audio pairs. This neural network consists of an audio and an image feature extraction branch, whose outputs are concatenated and fed into a dense network, to learn the essential correspondence between image and audio. I tested the quality of such pretraining by evaluating the audio feature extractor with environment sound classification task, and by evaluating the image feature extractor with ImageNet classification. I was able to show benefits of pretraining in both cases.

1. Introduction

I understand machine learning as a tool to extract information out of data just the way humans brain do, by imitating neurons firing, creating different pathways to understand the world through a plethora of data. The quest to uncover meaningful patterns from vast repositories of data has led to a burgeoning interest in audio-visual representation learning. We often see research in just audio or just image, but human interact with the world with both information, through our eyes and ears. I am particularly interested in extracting latent features embedded in videos. With more research aim to extract features in the absence of explicit supervision, I would like to explore unsupervised methodologies stems from the inherent challenges posed by real-world data, which frequently lack comprehensive labels necessary for traditional supervised learning approaches.

Among these methodologies, one notable avenue of exploration is the "Audio-Visual Correspondence" (AVC) task, written by Arandjelovic et al. [1]. By capitalizing on positive correspondences between audio and visual elements while simultaneously discerning negative associations, models trained on the AVC task can distill rich feature representations from unlabeled data.

In this project, I aim to delve into the foundational work of Arandjelovic et al. and elucidate the efficacy of AVC-based feature extraction in enabling downstream tasks. Specifically, I implement the exact architecture and downstream tasks (audio classification and image classification) as presented in the paper from scratch, in order to learn about image-audio pretraining, and validate the effectiveness of such approach. Additionally, time permitting, we aspire to extend our investigation to encompass audio-visual challenges such as video classification, thereby offering a comprehensive evaluation of the utility of AVC-derived features across diverse domains.

Through this endeavor, we aspire to contribute to the growing body of knowledge surrounding audio-visual representation learning, while also providing practical insights into the applicability of AVC-based feature extraction in real-world machine learning scenarios.

2. Background

The AVC task is a machine learning (ML) pretrain learning objective that takes in corresponding visual and audio inputs, to learn fundamental features of say audio and image frames. This is inspired by how human learn: if one sees and hears many videos of someone playing a piano, one should be able to understand and remember what a piano looks like and how it sounds. AVC essentially is a binary classification task that takes in both an image frame and an audio clip, and outputs whether these two inputs are related. Image and audio taken from the same video would have label "1" and pairs taken from different videos would have label "0".

Arandjelovic et al. [1] proposed an architecture to prove

the effectiveness of AVC as a pretrain objective. They designed a two-branch neural network that takes in three-channel image and log spectrogram respectively. The image and audio subnetwork share similar convolutional neural structure. The latent features are then concatenated and pass through several layers of dense layer to classify whether the image frame and audio clip is from the same source. They pretrained with 400k 10 second videos for two days, using 16 GPUs in parallel.

Pretrained results were evaluated by evaluating the audio and image subnetwork separately. To evaluate audio, they finetune the branch on an environmental sound classification task called ESC-50 [5]. To evaluate image, they finetune the image branch on ImageNet to perform image classification. In both cases, [1] showed superior performance of AVC pretrained neural network from the same structure with randomly initialized weights. They also showed superior performance against many baseline popular in both tasks.

3. Details of the approach

3.1. Model

The purpose of this project is to implement [1]'s approach from scratch, on the dataset they proposed. The neural network is shown in Figure 1. The neural network consists of two subnetwork of similar VGG [8] structure: a concatenation of 4 blocks that each consist of two convolutional layers and a maxpooling layer. Note that after each convolutional layer, there is a batch normalization and ReLU nonlinearity layer.

After pretraining the architecture againt the AVC task, I evaluate the pretrained subnetworks on an audio classification and an image classification task. Audio classification is done on ESC-50 [5], which includes 50 classes of environmental sounds. The audio subnetwork, loaded with pretrained weights, is fed into two fully connected layers and a softmax that outputs 50 class. Similarly, image classification is done on ImageNet with 1000 classes. The visual subnetwork, after loaded with pretrained weights, is fed into a two-layer dense network and a softmax with 1000-channel output.

3.2. Dataset

In the original paper, the authors used Flickr-SoundNet [3] to pretrain their architecture. Flickr-SoundNet cosist over 2 million videos, but I was unable to download this dataset due to hardware restrictions and link failure on [3]'s official website. For practical reasons, I used a subset of flickr that I found online, which instead of video, consists of one image frame paired with corresponding audio. There is a total of 5000 data pairs. In order to increase the number of data for both corresponding audio-image pair and non-

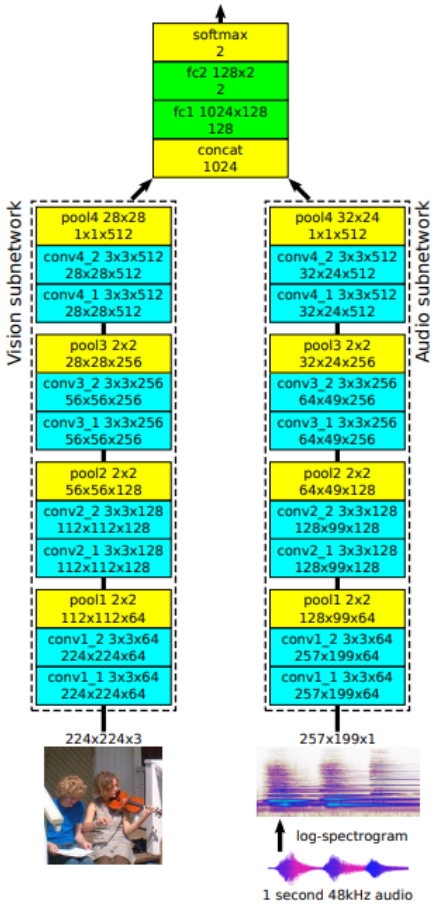


Figure 1. Designed architecture with implementation details. The prefix "conv, pool, fc, concat" means 2d convolution, 2d max pooling, fully connected, and concatenation. The numbers behind each of these layer represent the kernel size, and the numbers under the layers represent output dimension after the particulat layer.

corresponding audio-image pair, I implemented the following dataloader. For non-corresponding pair (the label being "0"), I randomly sample one image from the 5000 data entry, and randomly select 1 other entry and randomly sample 1 second audio from this entry. For corresponding pair (the label being "1"), I randomly sample one image from the 5000 data entry, and randomly sample 1 second from its corresponding entry. This way, I have enough data for pretraining.

For audio classification, I use ESC-50 [5], which includes 2000 wavforms with 5 second each. I randomly select 1 second during training, and use the first second for validation and testing. Training data and validation data is split 80:20 by default. The 1 second segments are then processed into log spectrogram to feed into the audio subnetwork.

For image classification, instead of using the entirety of

ImageNet [7], I only used the validation subset due to hardware restrictions. The validation subset includes 50000 images, which I then split 80:20 into training and validation subsets. The images are cropped into 224x224, horizontally flipped by chance, and fed into the visual subnetwork for training.

3.3. Implementation Details

Details for the architecture is described in Figure 1, which I will not reiterate. Note that the image from the input image has to be cropped to 224x224. 1 second audio has to be resampled to 48000Hz and processed into log spectrogram using the default pytorch implementation.

The output of both audio and visual subnetwork is of size 1x512, so the dense network for finetuning audio and image classification consists of a 512x128 dense layer, a ReLU nonlinearity, and a 128xnum_class dense layer.

The architecture was pretrained 24 hours on a RTX A6000 GPU, with batch size of 128 and 100 epochs. Audio classfication finetuning uses batch size of 64 and 50 epochs, and run for around 10 minutes. Image classification uses batch size of 256 and 50 epochs, and run for around 8 hours. All tasks were some variation of classification, so cross entropy loss is used for all tasks. Detailed code can be found in uploaded zip file or on GitHub [1]

3.4. Evaluation Metric

For each class in both audio and image classification, we can calculate the true positives (TP), false positives (FP), true negative (TN) and false negatives (FN) for sleep. Because this is a simple classification task, we evaluated standard accuracy, precision, recall, and F1-score as follows:

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

$$\text{Precision} = \frac{TP}{TP + FP}$$

$$\text{Recall} = \frac{TP}{TP + FN}$$

$$\text{F1-score} = \frac{2 * TP}{2 * TP + FP + FN}$$

In final evaluation, we look at macro averages for presision, recall, and F1.

4. Results

4.1. Pretrain

Figure 2 shows the loss history of training set and test set during pretraining. We see a continuous decrease in loss for training set but the test set has moderate decrease in loss and the loss increases after around 50 epochs.

[1] https://github.com/kaichieh121/ECE549

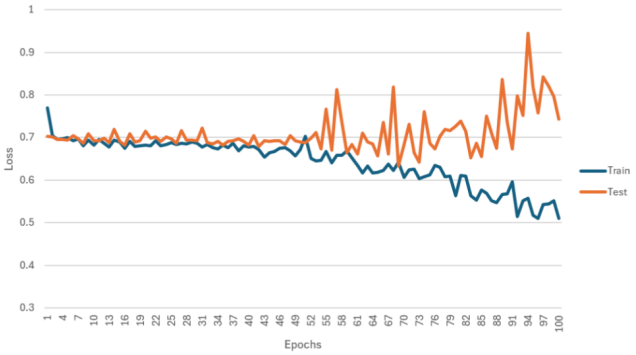


Figure 2. Loss history during pretraining

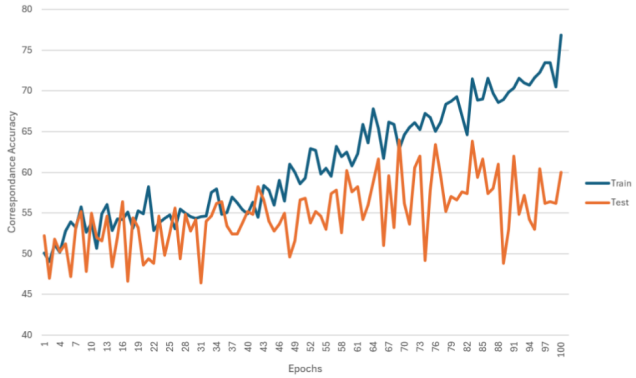


Figure 3. Accuracy history during pretraining. The accuracy corresponds to whether the architecture can correctly classify whether the given image-audio pair is from the same video.

Figure 3 shows how accuracy changes over the 100 epochs of pretraining. This accuracy is related to how well the architecture can identify whether an image-audio pair originates from the same video. We see that both trainng accuracy and testing accuracy increases, while test accuracy has more fluctuation in general.

4.2. Audio Classification

Figure 4 shows the loss changes during audio classification finetuning with randomly intialized weights. Both training and testing loss decreases fast in the first 15 epochs and flattens at different value (around 2 for test set and around 0 for training set).

Figure 5 shows the history of accuracy during audio classification training without loading pretrained weights. Both accuracy increases fast in the first 15 epochs and reaches different stable accuracy (100% for training set and 50% for testint dataset).

Similar trend is seen when conducting audio classification finetuning with pretrained weights, as shown in Figure

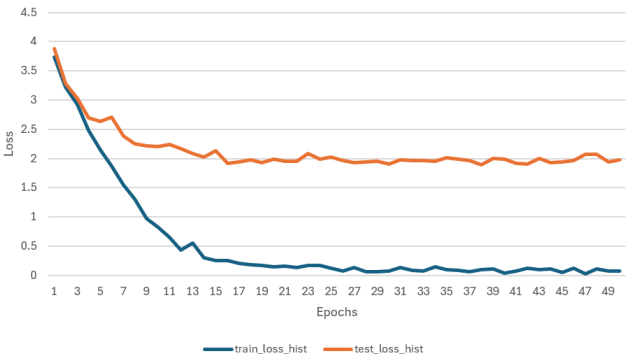


Figure 4. Loss history of audio classification with randomly initialized weights

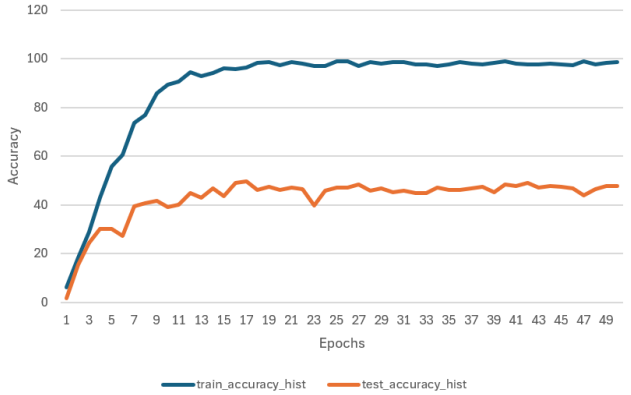


Figure 5. Accuracy history of audio classification with randomly initialized weights

6. In Figure 6a we observe a lowering of loss for both training and test set, but this time the test loss can reach a lower 1.5 as compared to the randomly initialized weights case. Similarly, in figure 6b accuracy for both sets increases while test accuracy was able to reach higher.

Table 1 shows the numeric evaluation results of audio classification with and without pretrained weights. We see that the pretrained architecture is able to outperform the randomly initialized architecture across the board on every single evaluation metric. The proposed model is able to outperform several baseline as denoted in the table. However, we see that although using the same architecture, my model is unable to outperform Arandjelovic [1].

4.3. Image Classification

Figure 7 and Figure 8 shows the loss and accuracy history of ImageNet classification. We see a much less significant decrease in loss as well as increase in accuracy for image classification, but the general trend is similar to audio classification. In both pretrained and randomly initialized cases, the loss and accuracy figures are very similar, so I only choose to show the pretrained versions.

Table 1. Results for audio classification. Accu, Prec, Rec stands for accuracy, presicion, and recall respectively. Mine random denotes my architecture finetuned with randomly initialized weights, and Mine pretrained denotes my architecture finetuned with pretrained weights.

Method	↑ Accu	↑ Prec	↑ Rec	↑ F1
SVM-MFCC [6]	0.40	-	-	-
Autoencoder [4]	0.40	-	-	-
Random Forest [6]	0.44	-	-	-
Arandjelovic [1]	**0.79**	-	-	-
Mine random	0.51	0.59	0.53	0.52
Mine pretrained	0.63	**0.65**	**0.63**	**0.61**

Table 2. Results for image classification. Accu, Prec, Rec stands for accuracy, presicion, and recall respectively. Mine random denotes my architecture finetuned with randomly initialized weights, and Mine pretrained denotes my architecture finetuned with pretrained weights.

Method	↑ Accu	↑ Prec	↑ Rec	↑ F1
Arandjelovic [1]	**0.32**	-	-	-
Mine random	0.14	0.15	0.14	0.13
Mine pretrained	0.15	**0.16**	**0.16**	**0.15**

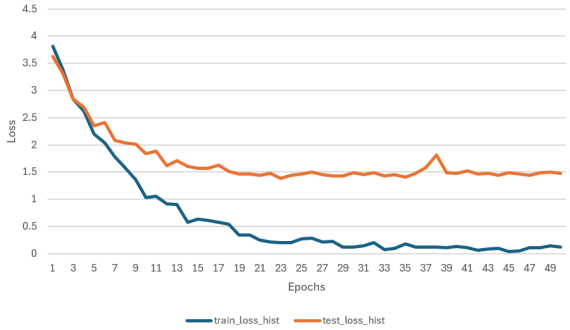
Table 2 shows the evaluation metric on the same architecture trained by [1], myself with randomly initialized weights, and myself with pretrained weights. We see that there is barely any difference when loading the pretrained weights, and my implementation significantly performs worse than the original paper.

5. Discussion

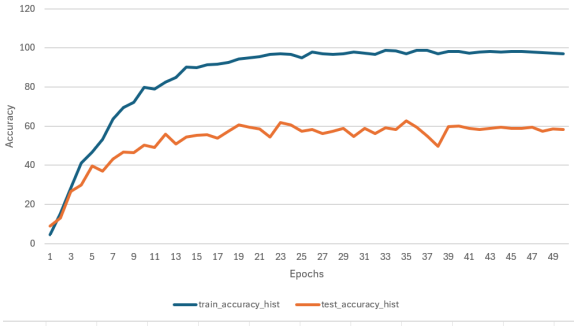
5.1. Pretraining

Careful examination of Figure 3 we see that the model was able to distinguish matching and non-matching audio-image pair for as high as 77% in the training dataset, and 64% in the testing dataset. This shows the model is able to capture some relationship between audio and image. This implies the subnetwork has learned some methods to extract latent audio and image features. However, when we look at how the testing loss increase significantly after epoch 50 as compared to training loss continuously decrease after epoch, it seems like the model overfitted on the problem and started to memorizing the pairings in the dataset.

I suspect the training dataset is too small. In future work, one should take in perhaps the entirety of Flickr-SoundNet or to use multiple image frames instead of just one image



(a) Loss history



(b) Accuracy history

Figure 6. History of loss and accuracy for audio classification with pretrained weights

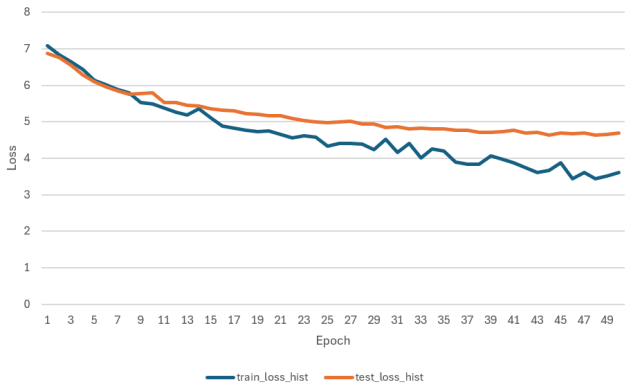


Figure 7. Loss history of image classification

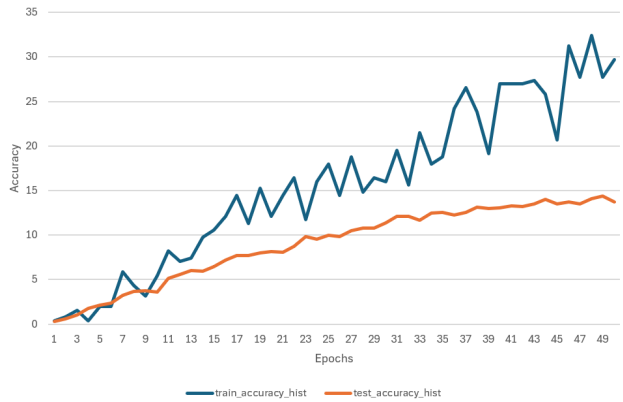


Figure 8. Accuracy history of image classification

pair to an audio wavform. By increasing the dataset, we will likely solve the overfitting problem. [1] uses 16 GPUs and trained for 2 days, so perhaps the issue with my pretraining is purely hardware/batch size dependent.

5.2. Audio Classification

To determine whether useful audio feature extractor was trained during pretraining, we look at environmental audio classification task specified in ESC-50. From the last two rows in Table 1, we see that loading the pretrained weights onto the audio branch allow the model to achieve higher accuracy, precision, recall, and F1 across the board, compared to the same model with randomly initialized weights, trained under the same hyperparameters. This shows pretraining under the AVC task allow the subnetworks to learn useful audio latent features. Our model outperforms three standard baselines on ESC-50 [2] [6]. However we did not out perform [1], although they use the same architecture and pretraining objective. The natural reason is simply they were able to use better hardware to pretrain on larger and more diverse Flickr-SoundNet. Since we were able to show AVC's superior performance against randomly initialized model, our implementation is also successful in a lesser degree.

Apart from pretraining the model using the full Flickr dataset, I believe we can improve the model by doing the following. Instead of log spectrogram, one can experiment with mel spectrogram, which generally lead to faster training and superior performance in my experience. Audio waveform fluctuates fairly quickly, so there could be benefits if we can include time information during training. For example, instead of using mismatched pair of audio and image frame from different entries, we can try using mismatched frame and 1-second audio of the same video. This could possibly allow the model to learn even better audio feature extractor.

5.3. Image Classification

Table 2 shows there is barely any improvements when finetune ImageNet image classification task using pretrained weights compared to randomly initialized weights. This is not a desired result as we expect the image subnetwork to learn image latent features as well as as audio as

shown in Table 1. Additionally, our implementation was unable to achieve anywhere as close to the paper that our design is based on.

The reason can be 2-fold. First, during pretraining, only one image frame is used for each entry, there lacks variety in image for the visual subnetwork to extract visual information with only 5000 images. If there is at least 5 second of video at 24 frames per second, we will have 120 times more images for training. Comparing this to audio, which has 5 to 10 times more segments per entry than image, I suspect the model only extract minimal image features. Secondly, we were only able to fit train the image classification task using only 50000 images due to hardware limitations. If we can train on the whole ImageNet dataset, all performance would increase. Perhaps the difference between randomly initialized weights and pretrain weights would be amplified. This experiment would take at least 300 hours (3 hours per epoch, 50 epochs, at least 2 experiments), which is not practical.

Future work can focus on scaling up the datasets for both pretraining and finetuning to improve image feature extraction, apart from all suggestions discussed in previous sections.

6. Conclusion

In conclusion, I was able to implement almost the entirety of [1] from scratch on available hardware. With limited access to pretraining dataset and limited hardware to run the full ImageNet finetuning, I was not able to show meaningful results when evaluating the pretrained weights on image classification. However, we can confirm the success of pretraining from the increase in pretraining accuracy. We can also confirm the usefulness of pretrained weights when loaded onto the audio branch finetuned for audio classification. This proves that AVC, while not having any label, can let a neural network distinguish audio and image pairs by learning essential audio and visual latent features.

7. Statement of individual contribution

I did everything from topic choosing, code, to report writing. Atharva Naik setup our initial meeting and did not show up. Later he dropped the course.

References

[1] Relja Arandjelović and Andrew Zisserman. Look, listen and learn, 2017. 1, 2, 4, 5, 6

[2] Relja Arandjelović and Andrew Zisserman. Objects that sound, 2018. 5

[3] Yusuf Aytar, Carl Vondrick, and Antonio Torralba. Soundnet: Learning sound representations from unlabeled video, 2016. 2

[4] Yusuf Aytar, Carl Vondrick, and Antonio Torralba. Soundnet: Learning sound representations from unlabeled video, 2016. 4

[5] Karol J. Piczak. Esc: Dataset for environmental sound classification. In *Proceedings of the 23rd ACM International Conference on Multimedia*, MM '15, page 1015–1018, New York, NY, USA, 2015. Association for Computing Machinery. 2

[6] Karol J. Piczak. Esc: Dataset for environmental sound classification. In *Proceedings of the 23rd ACM International Conference on Multimedia*, MM '15, page 1015–1018, New York, NY, USA, 2015. Association for Computing Machinery. 4, 5

[7] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. Imagenet large scale visual recognition challenge, 2015. 3

[8] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition, 2015. 2