An Exploration and Implementation of "Learning to Separate Object Sounds by Watching Unlabelled Video"

Luke Rowe
University of Victoria
lukerowe@uvic.ca

Quinton Yong
University of Victoria
quintonyong@uvic.ca

Jingjing Zhu
University of Victoria
jzhu@uvic.ca

Abstract

Our project is based on the paper "Learning to Separate Object sounds by Watching Unlabeled Video"[1] by RH.Gao et. al. . [1] proposes an innovative approach to audio source separation of unlabelled videos by capitalizing on the visual aspect of the video's frames then using a deep multi-label multi-instance learning network to learn audio frequency bases associated with each object class. Our contributions to the original paper's code base include the following: implement the data extraction, preprocessing, basis disentanglement, and post-processing stages (which are not provided in the original paper's code base), upgrade the network to support modern PyTorch [2], and modify / fine tune the network to use 4 classes of musical instruments and to perform well with our reduced dataset.

1. Introduction

This project builds upon the audio-source separation method proposed in [1]. The original paper's code base contains only the implementation of the Deep MIML network. Given this, and with consideration of the time constraints for this project, we have three main goals. The first goal is to implement the data extraction, preprocessing, basis disentanglement and post-processing stages of the proposed method, which are not included in the original paper's code base. The motivation behind completing these stages is that we are able to evaluate the performance of the audio-source separation. Our next goal is to tune the model architecture to obtain optimized results with a reduced dataset of 4000 samples with the following 4 instrument classes: drum, acoustic guitar, piano and violin. We include dropout layers within the Deep MIML network to reduce overfitting, we optimize the preprocessing image classifier to enhance the noisy "ground truth" labels, and we tune the hyperparameters to suit the reduced dataset. The reason for the size reduction of the dataset is that given the time restriction, it is infeasible to extract and preprocess the entire dataset suggested in [1]. The final goal is to run the full pipeline using the reduced dataset and to witness results similar in performance to the results provided in [1].

2. Contributions of the Original Paper

This paper [1] by RH.Gao et. al. proposes an innovative approach to audio source separation of unlabelled videos; the key insight to this approach is that providing additional visual context to unlabelled audio signals can assist in performing the audio separation task. This insight is justified in that the human auditory system can better discern the distinct sounds from an audio source when visual cues "promote" the segregation of sound.

The method proposed in [1] uses a Resnet-152 network pretrained on ImageNet [3] to detect objects present in each video. Also, the method performs NMF factorization on the magnitude spectrogram of each audio signal to obtain audio basis vectors that capture distinct audio spectral patterns. The proposed method uses a Deep Multi-Instance Multi-Label (MIML) network to learn the associations between the audio basis vectors and the object labels. Namely, the MIML network learns to accurately make a bag-level multi-label prediction for each input video. The Resnet-152 labels are employed to guide this association in a "self-supervised" manner; the term "self-supervised" is accurate since the network does not have access to any ground-truth labels, but instead predicts the objects present in each video with noisy Resnet-152 labels. Once the MIML network is trained, the method extracts the audio basis vectors that strongly correlate to each of the object labels. For each object label, its corresponding set of audio basis vectors unveils the key spectral patterns that define one objects' sound from the other. Finally, with a novel test video, the proposed method uses the learned audio basis vectors to reconstruct separated audio sources for each detected object in the video.

The main contribution in [1] is the new proposed method for audio source separation as described above. Concretely, the method employs both audio and visual information to guide the audio separation task without access to any ground-truth labels. Additionally, different from other papers [4, 5, 6, 7] which use artificially created videos, the method in [1] is the first to study audio source separation from a large scale of raw online videos which each contain multiple objects of differing sounds.

Architecture	Center Crop		Random Horizontal Flip		Random Crop	
	Un-normalized	Normalized	Un-normalized	Normalized	Un-normalized	Normalized
Max Pool	50%	64%	51%	63%	53%	64%
Sum	43%	58%	44%	60%	44%	59%

Table 1: Image classification accuracies with various architectures (2000 samples per class)

3.1. Preprocessing

For the preprocessing stage, we first extract the raw 10-second audio and video clips from each data sample. We then perform an image-classifier architecture experiment to increase the accuracy of the object-level labelling of the extracted frames. Lastly, we implement the audio basis extraction stage of preprocessing to extract the audio basis vectors from each audio magnitude spectrogram.

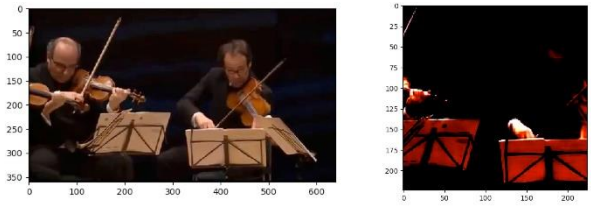

Figure 1: Illustration of original video frame vs image after Resize (256 x 256) → Center Crop (224 x 224) → Normalize

3.1.1. Dataset and Data Extraction

The training, validation, and test data we use was collected from Audio Set [8] which is a large set of links to different YouTube videos. More specifically, it consists of two CSV files namely "unbalanced data" (with 2.1 million links) and "balanced data" (with 22 thousand links) which we use for training / validation and testing respectively. Each video entry contains two timestamps, 10 seconds apart, representing a short video clip and a set of audio class labels indicating what object sounds appear in the video. Because the network works with unlabelled data, we discard the labels when training.

Part of the effort of recreating the paper is reconstructing the AudioSet h5 data which is not provided with the paper. Due to time constraints, we extract approximately 4000 video samples per audio class and our network supports the classification of 4 different classes: piano, violin, guitar, and drum. Note that due to some video samples being removed or copyrighted, approximately 200 samples per class are unavailable. The data extraction is performed using the command line program *youtube-dl* [9] wrapped in the library *FFmpeg* [10] to speed up the video's MP4 file download and to extract the WAV file. Finally, *OpenCV* [11] was used to extract 10 frames (1 frame per second) from the MP4 file needed for object classification.

3.1.2. Visual Frames

To support the unlabelled learning ability of the network, we use the ResNet-152 network pretrained on ImageNet on the 10 frames extracted from the MP4 file to predict the objects that appear in the YouTube video. A max pool is performed on the scores of each frame to generalize 10 frames into 1, and the labels with the top scores that are above a certain threshold are the classified labels for the video. These labels are then passed into the MIML network.

However, several issues are encountered when analyzing the label data from the frame images extracted from the videos. Some of these issues include the following: the objects (instruments) are not portrayed at an ideal orientation for classification, objects are concealed or out of frame, and miscellaneous noise (such as random text) covers most of the object. In particular, approximately 40 to 50% of the videos are mislabeled or under-labeled. We hypothesized that this mislabelling could greatly hinder our source separation performance and contribute to the overall loss.

Therefore, to attempt to fix these issues, we test several different slightly modified image classification architectures. Our initial architecture experiments also contain different pretrained networks, but it is clear from [12] that ResNet 152 has the greatest image classification performance in PyTorch. The following are the tested modified architectures ("Resize" is a resize to 256x256 and "Center Crop" is a center crop to 224x224): "Resize → Center Crop → ResNet → Max Pool", "Resize → Center Crop → Random Horizontal Flip → ResNet → Max Pool", "Resize → Random Crop → ResNet → Max Pool", "The previous architectures with Normalization before passing to ResNet", and "All of the previous architectures → Replace Max Pool with Sum over the 10 frames' labels"

We evaluate the performance of the architectures according to the "ground-truth" AudioSet labels for each of the 4 object classes. Note that "ground-truth" in this case is used loosely because the AudioSet labels correspond to the objects sounds that appear in the video and not necessarily

to the objects that appear in each selected frame. Despite this possible source of evaluation error, we can still determine the relative performance of these architectures by comparing the percentage of correctly returned labels. Table 1 illustrates the quantitative results of our experimentation with 2000 samples per class classified through each architecture. From Table 1, we observe that max pooling over frame scores and adding normalization gives the best classification performance. Based on these results, the modification chosen for preprocessing is the "Resize → Center Crop → Normalize → ResNet→ Max Pool" architecture (illustrated in Figure 1).

3.1.3. Audio Basis Extraction

To obtain the audio features, we extract audio basis vectors from each audio signal, which are then passed into the MIML network for training. Each basis vector captures a certain spectral pattern in its corresponding audio signal. Paired with the Resnet-152 labels, the MIML network can learn to associate these spectral patterns to the instrument labels. For each sample i, we first convert the WAV file for i into a time series representation at a sampling rate of 48000 Hz. We then convert the time series representation into a magnitude spectrogram $V_i \in M^{FxN}$, where F is the number of frequency bins and N is the number of time frames for each sample, by taking the Short-Time-Fourier-Transform (STFT) of the audio time series. V_i is then approximated by two matrices $V_i \approx W_i H_i$ using Non-negative Matrix Factorization (NMF) with Kullback-Leibler (KL) beta-loss and a multiplicative-update solver, as specified in [1]. Each column of W is an audio basis vector for its corresponding sample. Thus, the columns of W for each sample are then passed into the MIML network for training. We follow the implementation details as specified in [1]; we use *scikit-learn*'s [13] NMF implementation, and *librosa*'s [14] implementation of STFT to extract the audio basis vectors.

3.2. Per-Object Basis Disentanglement

The purpose of the MIML network is to learn the associations between the audio basis vectors and the object labels in a "self-supervised" manner. However, the MIML network itself does not create the list of basis vectors that define each object class. Following the training of the network, we extract the basis vectors that most strongly correlate to each of the object class labels; this process is referred to as per-object basis disentanglement. We use the normalized audio basis vectors from the training set for this

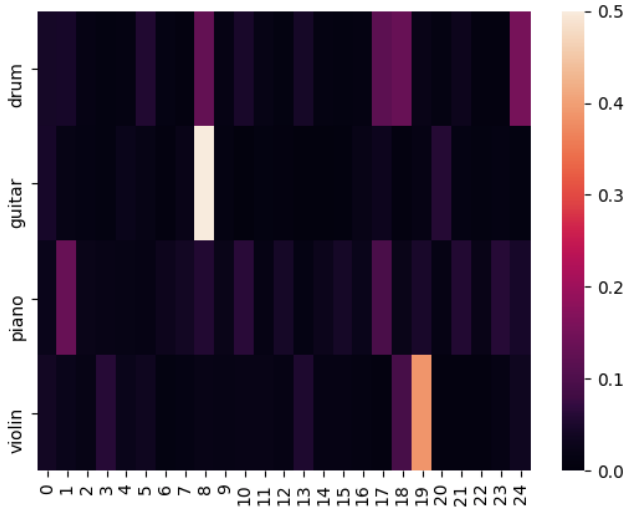

Figure 2: Basis vector to object relation map take from the output of the sub-concept pooling layer and softmaxed along the basis vector dimension

procedure. We pass these normalized basis vectors through the forward pass of the trained network up to and including the first max-pooling layer. We then softmax the output along the basis dimension M to get normalized object-matching scores along each of the M dimensions. The output of this "half-forward" pass will produce a relation map between the audio basis vectors and the class labels for each sample, as shown in Figure 2. For each audio sample, we can interpolate its relation map to find basis vectors that strongly correlate to a particular object class. As specified in Supp. [15], for each object, we only collect basis vectors that trigger the correct object-label prediction, and we only collect a basis vector if several other basis vectors for that sample trigger the correct object-label prediction.

3.3. Post-processing

Given we have disentangled basis vectors for each object class, we can reconstruct the separated audio source signals for each detected object in an unseen test video. For a novel 10-second test video, we follow the same object detection procedure as done in preprocessing to determine the object-label predictions. We also obtain the audio magnitude spectrogram V in the same way as preprocessing. For each of the j detected objects, we then concatenate these objects' corresponding basis vectors into one block matrix W as follows: $W = [W^{(1)}W^{(2)} \dots W^{(j)}]$

Further, we must approximate the activation block matrix H for V in order to reconstruct the audio magnitude spectrograms for each detected object in the video. We approximate H by performing NMF with a fixed W. A

challenge that we encountered in implementing this fixed-W NMF procedure is that the *scikit-learn* library is only able to decompose a matrix $C \approx \tilde{C} = AB$ with a fixed B. To circumvent this problem, we instead performed NMF on V^T, and approximated the H^T by keeping W^T fixed in the NMF procedure. We then obtain H by taking the transpose of the approximated result of the NMF procedure as follows: $H = (H^T)^T$.

Finally, we obtain an approximation $V^{(q)} = W^{(q)}H^{(q)}$ to the audio magnitude spectrogram for each detected class label q. As specified in [1], to obtain the separated source signal for each object q, we then soft-mask the mixture spectrogram $\mathbb{V}$ to obtain a spectrogram $\mathbb{V}^{(q)}$ that contains both magnitude and phase; we then perform an Inverse STFT (ISTFT) to reconstruct the separated source signal. As a result, we have a separated audio source signal for each Resnet-detected object in the novel test video.

3.4 Further Improvements

The following are additional enhancements that we make to the MIML network code base provided by [1] that optimize the performance on our dataset.

3.4.1 Upgrade to Modern PyTorch

The original paper's code base supports a previous version of PyTorch. Thus, we update several outdated commands to support modern PyTorch compatibility.

3.4.2 Tuned Hyper-Parameters

Since we have a reduced dataset, the default parameters of the MIML network do not lead to optimized results. We set the learning rate from 1e-3 to 1e-5 to reduce the rate in which the loss curve decreases. We also set the number of fully connected layers in the Siamese network to 3 instead of 1. Further, [1] does not specify hyperparameters for the per-object basis disentanglement stage of the algorithm. Empirically, we observe that when each class has roughly the same number of audio basis vectors, the audio-source separation is more convincing. Moreover, we observe that a smaller amount of audio basis vectors per class leads to improved audio-separation results. With consideration for these two observations, we set the per-object disentanglement hyperparameters as follows: for each object, we only extract a basis vector if the softmax score for that object is at least 0.15 and if there are at least 4 audio basis vectors with softmax score 0.1 for that particular sample. We also cap the number of basis vectors per class at 50.

3.4.3 Dropout

Although tuning hyper-parameters yields improved results with respect to test loss, test accuracy, and audio evaluation, we observe a substantial amount of overfitting from analyzing the training and validation curves. This is likely due to the reduced size in the data we use to train our network. Thus, we improve the network by adding dropout nodes into the model, as initially suggested in [16]. First, we experiment with a single dropout layer with a dropout rate of 0.2 on the input layer only, but this does not result in improved test performance. As our next attempt, we add a dropout layer with a lower dropout rate after each 1x1 convolution and after each activation within the Siamese Network. We experiment with dropout rates between 0.1 and 0.2, and we witness a slight reduction in overfitting. Finally, a single dropout layer is added immediately prior to the output of the Siamese Network with a dropout rate of 0.5 and this results in further reduction of overfitting. We experiment with the range of dropout rates in the range [0, 0.5], similar to [17], and with our dataset, the dropout configuration that yields the best test performance is the following: "A two dropout layers within each fully connected layer (specifically in the order Conv → Dropout → BatchNorm → ReLU → Dropout) with a dropout rate of 0.1 along with a dropout layer with dropout rate 0.5 immediately before the output of the Siamese Network".

3.5 Results

We show the results for two different configurations of the MIML Network. Specifically, we show the results without dropout layers and with dropout layers; we refer to these models as model 1 and model 2 respectively.

For model 1, we set all the dropout rates to 0, and we train the network with a batch size of 256, a learning rate of 1e-5, and 3 fully connected layers in the Siamese network. All other configurations were set to the default configurations as specified in [1]. The train and validation curves for model 1 are shown in Figure 3 and the test accuracy and test loss is shown in the 1st column of Table 2. From the model's training and validation curves, it is clear that the model significantly overfits to the training data. Even with the decrease in learning rate from 1e-3 to 1e-5, the validation loss stops decreasing after only about 150000 iterations, or about 12 epochs. Similarly, the validation mAP starts to decrease after about 12 epochs,

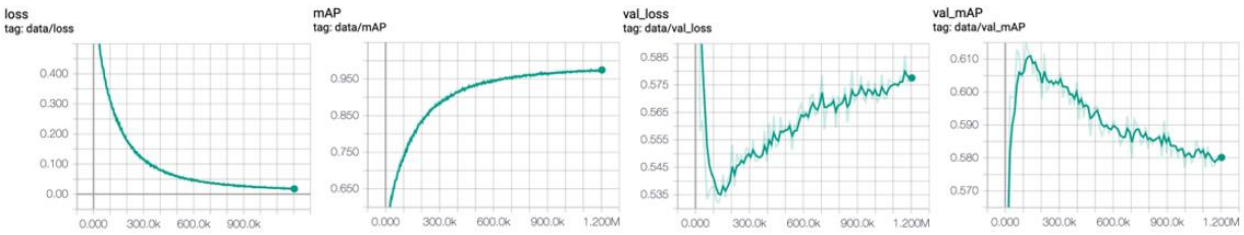

Figure 3: Training and validation curves for "Model 1: No Dropout"

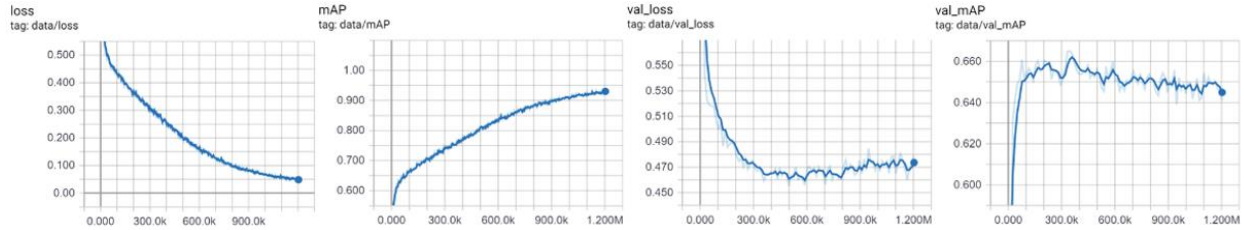

Figure 4: Training and validation curves for "Model 2: With Dropout"

despite that the training mAP continues to increase throughout the 100 epochs. These results confirm the need for a tweak in the architectural design of the network to mitigate the effects of overfitting.

For model 2, we set the dropouts according to the optimal dropout configurations as specified in 3.4.3. All other configurations are identical to the configurations set in model 1. The training and validation curves for model 2 are shown in Figure 4, and the test accuracy and test loss are shown in the 2nd column of Table 2. From the validation and loss curves, we observe that the overfitting is reduced significantly by adding the dropout layers. Even though the training mAP remains fairly consistent between the two models, the validation mAP improves by 5-7 with the addition of the dropout layers.

It is clear that the addition of the hidden dropout layers improves the performance of model overall. Namely, we witness a 14% increase in test performance from model 1 to model 2, and the validation mAP curve is subject to less overfitting. Empirically, we also observe an increased performance in the audio source separation task for model 2 (the audio results are attached in the GitLab folder). Using model 2, the reconstructed audio sources for each object class is able to better suppress the sounds produced by the other object classes. Overall, the reconstructed audio sources show clear signs of source separation, but the results do not compete with the results provided in [1]. From analyzing the reconstructed audio sources, it is clear that the network is able to capture the dominant spectral patterns of each of the four object classes. However, likely due to the noisy ResNet labels and lack of training data, the sounds of other object classes are still somewhat noticeable in the isolated audio sources.

	No Dropout	Dropout
Test Accuracy	42.15%	56.32%
Test Loss	0.6455	0.5457

Table 2: Test results of network with and without dropout

4. Discussion and Future Directions

A noticeable weakness in [1]'s model is that 40 to 50% of the training videos is a hindrance to the performance of the audio source separation. To test this hypothesis, we construct an experimental dataset that contains the extracted bases vectors from each video (as done previously) but instead include hard-coded label values corresponding to the audio source objects label provided by AudioSet. When training on this data, we observe a significant decrease in validation loss and increase in validation mAP. Moreover, we empirically hear better-separated reconstructed audio signals with the true labeled data (the WAV file of these true label experiment results are included in our GitLab). Thus, we can conclude that the mislabeling of objects that appear in the YouTube videos decreases the performance of the network's audio source separation. A possible future modification to mitigate this issue would be to further modify the image classifying architecture to improve labeling or to construct an image classifying network specifically trained on YouTube video frames rather than on ImageNet (to potentially learn to identify classification features despite noisy frames).

Also, in the data extraction stage, preprocessing each video in AudioSet takes around 15 seconds; approximately 5 seconds is used to download the 10 second clip and approximately 7 seconds is used to compute the NMF factorization. These two processes are the primary

bottlenecks for the per-video preprocessing time, and this greatly reduces the amount of data we could feasibly extract within our given time frame.

Additionally, we substitute the Short Time Fourier Transform in the network with the Constant-Q Transform but we are unable to correctly tune the *librosa* CQT and ICQT library parameters to produce meaningful results. Despite this, the instrument classification strengths of CQT discussed in Fitzgerald, D. et al. [18] shows a promising application to our audio source separation.

Therefore, in the future, we would like to compare and contrast our current audio separated samples results with the final results obtained from having a higher accuracy image classifier, more data, and to experiment with other audio transforms such as CQT.

References

[1] R. Gao, R. Feris and K. Grauman, ".: Learning to Separate Object Sounds by Watching Unlabeled Video", ECCV, 2018.

[2] Pytorch.org. (2019). PyTorch. [online] Available at: https://pytorch.org/ [Accessed 18 Apr. 2019].

[3] He, K., Zhang, X., Ren, S., Sun, J, ".: Deep residual learning for image recognition", CVPR (2016)

[4] F´evotte, C., Bertin, N., Durrieu, J.L, ".: Nonnegative matrix factorization with the itakura-saito divergence: With application to music analysis. Neural computation", 2009

[5] Hyv¨arinen, A., Oja, E, ".: Independent component analysis: algorithms and applications. Neural networks" 2000

[6] Virtanen, T, ".: Monaural sound source separation by nonnegative matrix factorization with temporal continuity and sparseness criteria. IEEE transactions on audio, speech, and language processing", 2007

[7] Zibulevsky, M., Pearlmutter, B.A, ".: Blind source separation by sparse decomposition in a signal dictionary. Neural computation" 2001

[8] Gemmeke, J.F., Ellis, D.P., Freedman, D., Jansen, A., Lawrence, W., Moore, R.C., Plakal, M., Ritter, M, ".: Audio set: An ontology and human-labeled dataset for audio events.", ICASSP, 2017

[9] Ytdl-org.github.io. (2019). *youtube-dl*. [online] Available at: https://ytdl-org.github.io/youtube-dl/index.html [Accessed 18 Apr. 2019].

[10] Ffmpeg.org. (2019). *FFmpeg*. [online] Available at: https://ffmpeg.org/ [Accessed 18 Apr. 2019].

[11] Opencv.org. (2019). *OpenCV*. [online] Available at: https://opencv.org/ [Accessed 19 Apr. 2019].

[12] Pytorch.org. (2019). *torchvision.models — PyTorch master documentation*. [online] Available at: https://pytorch.org/docs/stable/torchvision/models.html?fbclid=IwAR2XQ_0EiHeEKL94dyIZeX4zjNjHaCyLphjkzcTCGeE_gUCb_QmMzkjk6mU [Accessed 19 Apr. 2019].

[13] Scikit-learn.org. (2019). *scikit-learn: machine learning in Python — scikit-learn 0.20.3 documentation*. [online] Available at: https://scikit-learn.org/stable/ [Accessed 19 Apr. 2019].

[14] Librosa.github.io. (2019). LibROSA — librosa 0.6.3 documentation. [online] Available at: https://librosa.github.io/librosa/ [Accessed 16 Apr. 2019].

[15] R. Gao, F. Feris and K. Grauman, *Learning to Separate Object Sounds by Watching Unlabeled Video (Supplementary Materials)*. 2018.

[16] G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. R. Salakhutdinov, "Improving neural networks by preventing coadaptation of feature detectors.", arXiv:1207.0580, 2012.

[17] S. Park and N. Kwak, "Analysis on the dropout effect in convolutional neural networks.", Asian Conference on Computer Vision Springer, 2016

[18] D. Fitzgerald et al., "Shifted NMF Using an Efficient Constant-Q Transform for Monaural Sound Source Separation", 2011.