Audio Classification Using Convolutional Neural Networks

Poet Larsen
plarsen2@wisc.edu

Reng Chiz Der
rder@wisc.edu

Noah Haselow
nhaselow@wisc.edu

Abstract

As audio becomes more central in modern technology, the need to quickly and automatically classify it becomes more demanding. While a person can accurately understand the contents of a sound clip relatively easily, their speed is limited by the clip's length and complexity, making their efforts ineffective at scale. Additionally, many security applications continue to demand better audio classification software for improved surveillance and intelligence gathering. Significant progress in audio classification can also improve accessibility for the deaf and blind in modern systems, which has clear importance for the creators of these systems and for the impaired themselves. Previous audio classification models focus on using traditional machine learning methods, e.g. KNN, or multi-layered perceptron models to classify audio data. Unfortunately these models are susceptible to overfitting and plateauing accuracy. In this paper we take a new, modern approach to audio classification, using a Gated-CNN architecture to classify audio spectrograms. This architecture has two convolutional layers, with each layer incorporating a sigmoid and linear activation function in parallel. After these layers, the data is sent through 3 fully connected layers and the output is computed using softmax regression. To train and test our model, we randomly sampled 10 classes from the Google Ontology AudioSet. This dataset represents YouTube audio clips assembled by a group of researchers at Google specifically for training neural networks on audio data. To convert these audio clips, they were run through software that converted them from audio files to spectrograms. We conclude this paper by comparing results between models and the potential for future work.

1. Introduction

Classifying objects through the use of computer vision has been a popular task for many deep learning researchers and industry programmers, finding increasing use in new problems such as automated cars and video security. Many researches and experiments have been done on image classification and high performance models have been reported and developed. Cementing themselves as powerful tools, many CNN architectures are beginning to see use in other areas of interest beyond image classification.

In the realm of audio classification, convolution architectures have been successfully applied in audio fields with less disturbance and noise such as speech and music analysis [10]. For instance, one of the state of the art automatic music tagging networks is a deep, Fully Convolutional Network (FCN), which was only recently introduced in 2016 [4]. This network achieved AUG of 0.851 (or 85.1% prediction accuracy) on 250,000 audio clips of 50 unique genres. However, convolutional networks are still in their infancy for Acoustic Scene Analysis (ASA), or Audio Event Detection(AED). Acoustic Scene Analysis is the task of analyzing sources of arbitrary acoustic events in a audio data [3]. ASA, or also known as classification of environmental sounds, is still primarily done using traditional methods including Gaussian mixture model, support vector machines and hidden Markov models to manually extract features such as mel-frequency cepstral coefficients [10].

In recent years, key requirements for developing deep, comprehensive neural networks for audio classification have begun to align in the deep learning research realm. In particular, the creation of the Google Ontology AudioSet[6]has helped open the gate to further audio classification research using Convolutional Neural Networks, with a comprehensive audioset available amongst researchers. Before this, many audio datasets were fairly small and there were few comprehensive collections shared amongst researchers as compared to the likes of image datasets.

To begin our research, we explored different benchmark models including VGG-16, VGG-19, AlexNet and ResNet to infer the aptness of our data processing and data-augmentation. Initial concerns arose from the transformation of audio clips to spectrograms due to the spatial nature and high variability of quality of sound recorded within each clip. Once we were certain our data was properly prepared, we explored different regular data-augmentation procedures by treating the T-F spectrograms as regular images and observed their performance through these baseline models. We then proposed using a Gated-CNN architecture

as our new model, finding its accuracy to be greater than all of the baseline models[13].

The paper is organized as follows. Section 2 introduces the related work associated with this task, completing a brief literature review. Section 3 discusses data preprocessing and our proposed main model. Section 4 explores different engineering techniques, including data-augmentation, CNN structures, Max-Pooling, Batch Normalization and Dropout in audio recognition. Lastly, section 5 and 6 summarize our results, discuss the meaning of the results, and propose future plans and options.

2. Related Work

Google AudioSet The image recognition task has achieved huge successes partly due to the ease of data collection. The wide availability of huge image dataset includes the ImageNet data set that contains more than 14 million clearly hand-annotated images of more than 5000 categories [5]. It has been the capstone for the huge development of Convolutional Neural Network (CNN) including the famous AlexNet. [12] [9] [7]. However, there was little such dataset available for audio data prior to 2016. Most of the audio datasets that were available back then are more limited and narrow scoped data including Sports-1M for sports video and ActivityNet for human activities [6]. However, the audio classification task and scene has greatly improved with the introduction of benchmark audio dataset such as YouTube-8M [2], a multi-label video classification dataset of approximately 8 million manually-annotated videos from YouTube and Google's latest benchmark large-scale audio set, AudioSet [6].

Models Back in 2015, Justin Salamon and Juan Bello noticed, from what limited research had been done on sound classification, that only a handful of papers actually focused on specific sound source in the scene and not the auditory scene type [11]. In their paper, they used the UrbanSound8k dataset which has around 9000 samples of up to 4 seconds duration clip each of 10 possible city environmental sound labels. Doing so made them pioneers of environmental sound classification and have provided us the idea of prepossessing audio clips into time-frequency representations. In their work, they implemented an unsupervised learning feature extractor with a spherical k-means algorithm. Although they only reported a testing accuracy of less than 75%, they obtained a 5% accuracy over their baselines, successfully capturing and proving the importance of temporal structure.

In 2015, Karol Piczak [10] introduced the usage of Convolutional Neural Networks in the audio classification of environmental and urban sound source task. Just as Justin Salamon et al., Karol used the UrbanSound8k dataset and performed training using CNN. He used 2 convolutional layers, max-pooling, ReLU activation functions and 2 fully-connected classification layers as one of the models that he trained. Although, despite his report that these presented baseline models were not fine-tuned and further evaluation on his CNN architecture was needed, he obtained close to a 5% improvement on prediction accuracy over leading models at the time.

Since then, many different CNN have been utilized for audio classification and the results have been discussed in [8]. One paper in patricular, published by Xu et al. [13], implemented a Gated Convolutional Neural Network. The main difference that they proposed is to replace the ReLU activation with learnable gated linear unit (GLU) and to implement localization method for audio events from the weakly labeled data. Because of their success in improving the testing accuracy over other baselines, this has become the reference and motivation behind our model.

3. Proposed Method

We propose using a Gated-CNN neural network architecture with 2x2 max-pooling, 3 fully connected linear layers, dropout after each convolutional layer with P=.5, and training for 50 epochs. This model takes inspiration from the Gated-CNN architecture found from Xu et al. paper, with additional modifications inspired by other audio classification papers, that will be discussed later.

4. Experiments

4.1. Dataset

We trained on the Google Audioset, a dataset consisting of 632 audio event classes and a collection of 2,084,320 human-labeled 10-second sound clips drawn from YouTube videos. The sound clips cover different categories and wide ranges of common everyday environment sounds (i.e. human sounds, animal sounds, music), each of which are further split into subclasses to form a hierarchy encompassing the 632 event classes.

4.1.1 Data Gathering and Preprocessing

To gather the audioclips, we used a script written by Alex Nichol published on GitHub (unixpickle/audioset) that takes the collection given to us by Google Audioset and downloads the videos. Unfortunately, this script takes about 4-5 seconds per video to run, which is caused by (1) the size of the audio clips, and (2) a purposeful small time-out between downloads to prevent possible blacklisting by YouTube due to downloading large quantities of audio. This places a heavy restriction on the amount of data we can use to train our model, since downloading all 2 million clips would take approximately 3-5 months of constant running

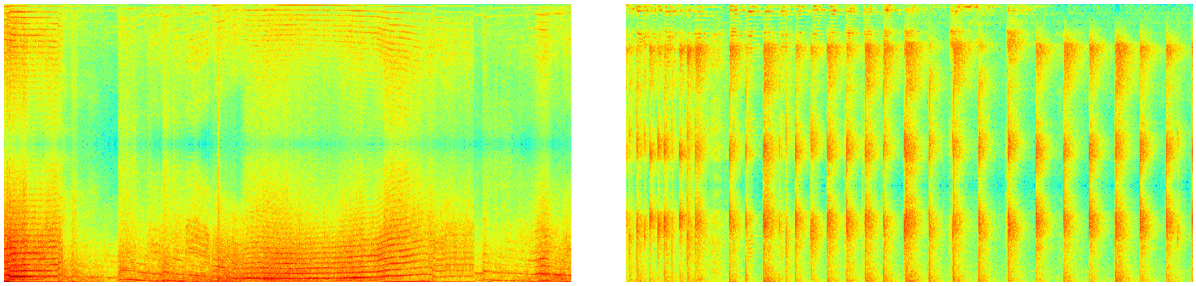

Figure 1. Example of spectrograms depicting a vehicle running (left) and hands clapping (right).

just to retrieve our data. To shorten this process, we selected 5,000 audio clips from 10 classes, which only took 6-12 hours to download, resulting in around 4,700 clips after various scraping errors (i.e. video taken down, copyright restrictions). In addition to the shortened download time, this strategy presented multiple other benefits:

1. Training time is similarly shorter with a smaller dataset. The training time of most of our models were in the span of hours rather than minutes, meaning that taking only 0.25% of the complete dataset vastly improved the speed we trained our model. We expected that even using a GPU, it would have taken over 10 hours *per epoch* to train a model with the full dataset, which isn't possible in the context of this class. Additionally, the shorter training time gave us more opportunities to fine-tune hyper-parameters and explore other models, which we felt was more beneficial to our learning as students.

2. Shrinking the number of classes that we trained on from 632 to 10 created a clearer distinction between the classes since we made sure to select classes from different areas of the ontology's hierarchy. Our efforts here were aided by the fact that the hierarchy has 7 top levels, so selecting 10 distinct classes was simple since we only needed a few overlapping labels. This produced better results in training because it's easier to distinguish between dissimilar sounds (whistling is much harder to distinguish from singing than it is from an explosion).

After receiving our data, we wrote scripts to unzip the files, clean our data, and convert the audio clips into spectrograms with 3 color channels. All of these scripts are available on GitHub (nhaselow/stat479). These scripts took about 0.75 - 1.25 seconds per clip to run (further justifying a smaller dataset), which is due to the size of the audio files and the complexity of the spectrograms. All of this scraping and cleaning results in about 4,700 spectrograms, each labelled as one of 10 unique audio classes in a separate csv file. This set of images was then further split into training, validation, and test sets following an 80/10/10 divide. The splitting was done randomly, according to the ratio of the data for each class. Anyone who intends on recreating our experiments should expect downloading and cleaning our data to take between 10 and 15 hours in total, increasing linearly if they attempt to use more data.

4.1.2 Understanding Spectrograms

One of our goals in this project is to reframe audio classification as an image classification problem, and spectrograms provide a great avenue to do so. A spectrogram is a visualization of spectral frequencies over time (see: Figure 1), created via a discrete short-time Fourier transform (the details of this method aren't relevant to our analysis). It involves three dimensions - time (x-axis), frequency (y-axis), and logarithmic amplitude (color) - which notably exclude the phase of the signal it represents. This means that spectrograms can't be used to recreate the original sound clip, as they aren't complete descriptions of audio. We aren't concerned about this loss of information because we believe that frequency and amplitude over time will be sufficient for accurately classifying audio.

4.2. Scripts and Software

All code that we wrote to scrape and clean our data can be found on Github (nhaselow/stat479). The repository contains a ReadMe file describing each script that we wrote and used, along with the scripts themselves. These scripts were written and have been tested in Python version 2.7.10. Other versions are untested. We also used the download function in Alex Nichol's AudioSet repository (unixpickle/audioset) to scrape the audio files as described earlier.

4.3. Architecture

In this section we will discuss the different architectural designs we made and the motivation for why. We will also discuss why we chose the Gated-CNN architecture as our main design, as well as the reasons for using

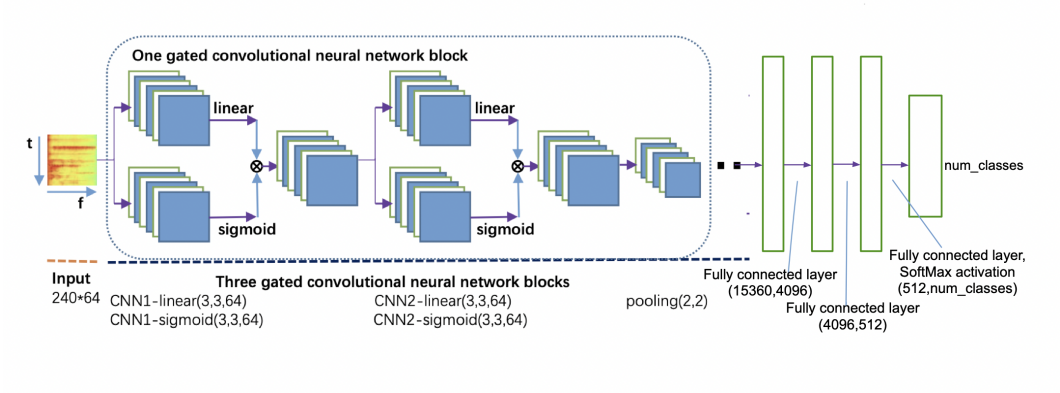

Figure 2. Gated Convolutional Neural Network with Multi-Layered Classification Model

additional engineering techniques from the initial architecture, including max-pooling adjustments, image augmentation, dropout, batch normalization, and binary classification. Through the use of these different techniques, we looked to find the best deep learning model to predict audio labels from spectrograms.

4.3.1 Transfer Learning

One of the reasons that we performed transfer learning was to give us baseline models to compare and act as benchmarks for the Gated-CNN model. Additionally, transfer learning also ensured that we had properly transformed our audio clips into appropriate spectrograms and that our data was properly connected with our coding environment, Google Colab. As we have discussed earlier, the transformation of spectrograms is not a standardized process and there are several parameters that can be fine-tuned.

For AlexNet, we did not make any make changes to the architecture and is similar to the one provided by Professor Sebastian Raschka. However, we note that the original AlexNet architecture was designed for 224*224*3 input and our spectrogram has 128*128*3 input. This subtle change required only a few numerical modifications to the size of the input going into the fully connected layers of the architecture. For Resnet34, no changes were made.

For VGG-16 and VGG-19, we did make changes to the classifier layers. Particularly, we changed the output layer by adding two additional output layers. With our setup, we note that VGG-19 does not produce better results that VGG-16 with additional computational time.

4.3.2 Gated-CNN Architecture

The motivation for using a Gated Convolutional architecture (Gated-CNN) came from Yong Xu*, Qiuqiang Kong*, Wenwu Wang, and Mark D. Plumbleys paper Large-Scale Weakly Supervised Audio Classification Using Gated Convolutional Neural Network. In their paper they highlight the benefit of using the sigmoid and linear functions stating, These GLUs can reduce the gradient vanishing problem for deep networks while retaining non-linear capabilities through the sigmoid operation. [13] As seen in the Gated-CNN architecture photo above, the linear activation function helps to dampen potential exploding/vanishing gradient problems arising from the use of the sigmoid activation function. This allows the model to learn a non-linear decision boundary, while also having linear-like properties during backpropogation. Additionally, the use of only 2 convolutional layers make the architecture relatively fast to train when compared to other models like VGG-16 and VGG-19. To simplify our architecture, we removed the Bi-Directional Recurrent Neural Network and Feed-Forward Neural Network from the original paper, for a simple Multi-Layered Perceptron model.

4.3.3 Max-Pooling

After finding a model we understood how to implement, and that had proven results classifying other spectrogram data, we looked for additional methods to try and improve the networks accuracy. Further literature review led us to Ossama Abdel-Hamid, Abdel-rahman Mohamed, Hui Jiang, and Gerald Penns paper Applying convolutional neural networks concepts to hybrid NN-HMM model for speech recognition [1]. In their paper they suggested that max-pooling had substantial benefits in CNN architectures for speech/audio recognition.

Because speech and audio data rely on a time element, there is often some level of auto-correlation and similarity between one slice of audio and its neighbors. By using max-pooling, the neural network can better generalize and reduce the noise that can be around the focal sound within a given kernel.

4.3.4 Image Augmentation

Due to limited resources and a finite amount of time, our collected data-set was relatively small. To mitigate this issue we used several image augmentation methods to further improve the accuracy of our model. Techniques we used included image rotation (P=0.5), image resizing, and pixel normalization. Image rotation allowed us to create more randomized images, increasing the noise of our training data and the robustness of the model out of sample. Resizing our images helped to prevent over-fitting and have the network focus more on general features of the spectrograms. With the images originally ranging from 1500x700 to 1000x700, there was concern that the model could overfit to certain features leading to good training results but poor out of sample accuracy. Lastly, pixel normalization was used to help speed up convergence and reduce the pull that some variables may have over others. This process standardizes all the input data, helping the network learn and prevent any one variable from having excessive pull on the models learning.

4.3.5 Dropout

Dropout is a useful and efficient tool in deep learning that helps mitigate potential over-fitting and the reliance on neurons in the network. By adding a dropout layer, with P=.5, after each convolutional layer, the network may be able to better generalize from our training data.

4.3.6 Batch Normalization

As an additional engineering tool in deep learning, batch normalization can help to speed up training and further reduce issues of vanishing gradients, a major concern when using the sigmoid activation function. Additionally, batch normalization is said to help reduce internal covariate shifts, leading to faster convergence. In our work we used it as another technique to tweak our architecture, however it did not appear to help with our training accuracy.

4.3.7 Binary Classification

To better measure the quality of our network, we also tested it over a binary class sample. To gather this sample we took the images we already had and arbitrarily split them in half, with class labels 0 and 1 having roughly 2500 images each. By creating new samples with fewer classes we have more data to test and train on, giving us more insight on the quality of the network. Additionally, we tested on both RGB images and grayscale to see if the reduction in parameters would help the network better generalize.

4.3.8 Color Modification

As mentioned above, we also manipulated the spectrogram dataset to have grayscale and RGB images. Since the color of our images only reflects the amplitude of the sound in the spectrogram, converting our images to grayscale might not negatively impact our results while drastically improving training time of our models; in fact, doing so may help our model better generalize outside of its training set. However, due to time constraints we were only able to test these changes on our binary classification dataset. Future tests would look to change our 10 class dataset from RGB to grayscale.

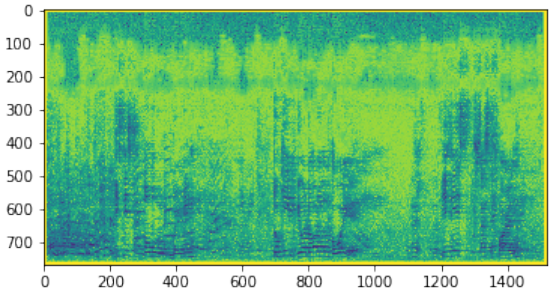

Figure 3. Example of a spectrogram converted to grayscale

5. Results and Discussion

After training our different Gated-CNN models, we found that the Gated-CNN architecture using 2x2 max-pooling, dropout with P=.5, and running for 25 epochs had the best testing accuracy (45.957%) when predicting images from 10 different classes. When compared to our transfer learning models, we find VGG16 and VGG19 having the next best results, with testing accuracy's of 43.19% and 42.13% respectively.

When testing on binary class labels we found that the Gated-CNN architecture using 2x2 max-pooling, dropout with P=.5, and running for 50 epochs had the best testing accuracy (65.106%). However, running the same model for 25 epochs had similar accuracy results (64.255%) suggesting that the 50 epoch model was either stuck in a local optima, or both were overfitting. Converting the images to grayscale for binary classification did not improve the accuracy of the results, suggesting that color may not be playing a major role in the classification process. However, it is difficult to fully conclude that image color doesnt have an effect on accuracy, as this was only run on the binary classification dataset and our dataset was small.

When testing the different engineering techniques discussed above, we only found batchnorm to not be effective in helping the model improve its accuracy or speed up convergence. However, we did find that image augmenta-

tion and dropout (P=.5) both moderately improved the accuracy of all models by roughly 1%-1.5% for both binary and multinomial classification. Furthermore, we found that increasing the size of max-pooling layers did not improve the accuracy of the Gated-CNN architecture. These results were the most surprising because Ossama Abdel-Hamids et al. paper found larger max-pooling layers, up to 5x5 kernel size, further improving the accuracy of their models. This did not end up being the case for our models. However, further research and training should be done to make definitive conclusions as our dataset was again quite small. The small size of our dataset may've also limited the effectiveness of batchnorm, leading to inconclusive results on its usefulness for our model.

Model	Training Accuracy(%)	Testing Accuracy(%)
VGG-16	84.61	43.19
VGG-19	81.21	42.13
AlexNet (15 epoch)	32.84	30.75
Resnet	82	31

Table 1: Results for benchmark models

Model	Variation	Training Accuracy(%)	Testing Accuracy(%)
Gated-CNN	2*2 Max-Pooling	89.7	40.1
Gated-CNN	4*4 Max-Pooling	94.7	32.3
Gated-CNN	BatchNorm(.5), 25 epochs, 2*2 Max-Pooling	70.8	45.957
Gated-CNN	Same as above, Binary Classification	65.0	64.3
Gated-CNN	Binary Classification, 50 epochs	70.8	65.1

Table 2: Results of different models

6. Conclusions

In conclusion, we found that the gated-CNN architecture using 2x2 pooling, dropout with a probability of .5, and running for 25 epochs had the best testing accuracy for 10 class classification, with the same model running for 50 epochs having the best binary classification accuracy. VGG16 and VGG19 had similar accuracy results for the 10 class classification problem. In addition to these results, we found that across the board, use of image augmentation and dropout did help increase the accuracy of all models by roughly 1%-1.5%. Furthermore, we found that further increasing the size of the max-pooling layers beyond 2x2 did not improve the accuracy of the model. Similarly batch normalization did not help improve the accuracy of our model nor did it improve the convergence speed. Despite these techniques not improving our accuracy, we believe that they could be helpful if we had more data.

As described in detail in Section 4.1.1, there were several limiting factors that restricted our ability to scrape a dataset of a sufficient size, including the large file size of our audio clips and the extensive preprocessing needed to convert them to spectrograms. While we utilized many image augmentation techniques trying to counteract this setback (image rotation and resizing, pixel normalization), our efforts couldn't fully replace the benefit of having a complete dataset. Even on a very small dataset, however, we were able to achieve a binary classification accuracy of over 65%, which suggest that the choices we made regarding the architecture of our model were well made.

6.1. Future direction

For these reasons, we suggest that the first priority of any future work be in scraping a larger dataset. The scripts we used to get our 5,000 examples extend seamlessly to larger sets, so the only resource spent in doing so should be time. We expect this task alone should increase the test accuracy of our models significantly, which will highlight even more benefits and deficiencies of our architectures. After doing so, future work can explore increasing the number of audio classes that the model can handle. We reduced the number of classes from 632 to 10, and ambitious researchers can expand our model to dozens more classes, or even the full 632.

Furthermore, future work should include more fine-tuning of our engineering tricks, including exploring different dropout probabilities, batchnorm usage, and further image augmentation to refine our model. Besides that, fine-tuning of the spectrogram can be done with the librosa library in python. With it, more experiments can be done on merely the transformation, including transforming the audio clip to Signal Analysis, MFCC, STFT, NFC etc. Future research may head toward training with raw audio file instead of converting them to spectrograms to reduce the data loss in the process.

7. Acknowledgements

The authors would like to thank Professor Sebastian Raschka for his guidance and valuable feedback.

8. Contributions

We met biweekly to organize our work, delegate tasks, and relay information about what we accomplished. During this time, we discussed preprocessing strategies, issues with data scraping efficiency, possible architectures, various literature, and much more, so many high-level parts of our project were a true team effort.

Noah led the preprocessing and data cleaning operations needed to capture and convert our audio files. Reng and Poet then together led all training, testing, and building of the different architectures discussed in this paper. All three of us took a heavy role in researching the necessary references and literature needed to help with completing the project.

References

[1] O. Abdel-Hamid, A.-r. Mohamed, H. Jiang, and G. Penn. Applying convolutional neural networks concepts to hybrid nn-hmm model for speech recognition. In *2012 IEEE international conference on Acoustics, speech and signal processing (ICASSP)*, pages 4277–4280. IEEE, 2012.

[2] S. Abu-El-Haija, N. Kothari, J. Lee, P. Natsev, G. Toderici, B. Varadarajan, and S. Vijayanarasimhan. Youtube-8m: A large-scale video classification benchmark. *arXiv preprint arXiv:1609.08675*, 2016.

[3] S. Burger, Q. Jin, P. F. Schulam, and F. Metze. Noisemes: Manual annotation of environmental noise in audio streams. 2012.

[4] K. Choi, G. Fazekas, and M. Sandler. Automatic tagging using deep convolutional neural networks. *arXiv preprint arXiv:1606.00298*, 2016.

[5] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

[6] 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)*, pages 776–780. IEEE, 2017.

[7] 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*, pages 770–778, 2016.

[8] S. Hershey, S. Chaudhuri, D. P. Ellis, J. F. Gemmeke, A. Jansen, R. C. Moore, M. Plakal, D. Platt, R. A. Saurous, B. Seybold, et al. Cnn architectures for large-scale audio classification. In *2017 ieee international conference on acoustics, speech and signal processing (icassp)*, pages 131–135. IEEE, 2017.

[9] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[10] K. J. Piczak. Environmental sound classification with convolutional neural networks. In *2015 IEEE 25th International Workshop on Machine Learning for Signal Processing (MLSP)*, pages 1–6, Sep. 2015.

[11] J. Salamon and J. P. Bello. Unsupervised feature learning for urban sound classification. In *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 171–175. IEEE, 2015.

[12] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1–9, 2015.

[13] Y. Xu, Q. Kong, W. Wang, and M. D. Plumbley. Large-scale weakly supervised audio classification using gated convolutional neural network. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 121–125. IEEE, 2018.