Explainable CNN and Spectrogram Comparison for Musical Instrument Classification

Chan, Zachary
School of Electrical and Computer Engineering
Georgia Institute of Technology
Atlanta, Georgia 30332
zchan9@gatech.edu

Vats, Tavish
School of Electrical and Computer Engineering
Georgia Institute of Technology
Atlanta, Georgia 30332
tvats3@gatech.edu

I. Task

The goal of the project was to develop a convolutional neural network (CNN) for the multi-label classification of musical instruments in polyphonic recordings. Therefore, CNN should identify classes of musical instruments in a song that contains many instruments that are playing at the same time. Instrument identification from polyphonic audio is considered difficult, even with state-of-the-art research. The motivation was not to beat the state of the art, but rather to explore using CNN architectures for instrument classification and using gradient-based explainability to determine why the network makes its decisions. This demonstrates the effectiveness of image processing techniques for audio processing. Instrument identification in music has applications in automatic music tagging. For example, a music-sharing website such as Spotify or Youtube can have tags on each song that can be used for search engine optimization.

II. Literature Survey

A CNN architecture was motivated by discovering prior success with using CNNs for instrument classification and other audio classification tasks [1] [2]. The use of recurrent neural networks (RNN) has also been seen in audio classification research because audio is naturally time-series data. The combination of CNNs and RNNs in various ways has been shown to improve the accuracy of audio classification compared to CNN or RNN alone [2]. The goal of the experiment was not to create the best-performing model. Rather, it was to compare spectrogram features using visual explainability. RNNs are not as easily explainable compared to CNNs, so the experiment used only a CNN architecture. The input to CNNs is typically two dimensional, however, audio is one-dimensional. In order to use a CNN on audio, a time-frequency representation of the audio was used. The most commonly used representation is the Mel spectrogram because it is shown to train classifiers more effectively than the standard STFT spectrogram. Some researchers have used other transforms like Chroma and Constant Q transform (CQT), however, it was decided to only use Mel spectrograms as they mimic the human ear's perception of sound more accurately [3]. The explainability of CNN's decision-making was also of interest [4]. The CNN architecture warrants the use of post hoc explainability. Post hoc explainability is a paradigm in explainable artificial intelligence that aims to explain a model's decisions after training has been completed [5]. Gradient-based methods have been common in research, and one effective method is gradient-weighted class activation maps (GradCAM) [6]. GradCAM is used to explain why a network makes a particular decision by highlighting features of the input image that most strongly contribute to its label decisions. Deep Feature Factorization is a visualization method that segments an input image into similar 'concepts'. It reveals groups of pixels that a CNN believes belong to similar target labels [7]. Both of these methods were used to visually explain the CNNs.

III. Approach

A. Dataset

Creating a network from scratch can be time-consuming because it takes experimentation to design. Therefore, pretrained networks were used and transfer learned to the OpenMic2018 dataset [8]. The OpenMic2018 dataset contains 20,000 10-second polyphonic audio recordings of instruments with associated labels of which instruments are present. The files are stored in .ogg format and there are 20 instrument labels. Fig. 1 below shows a statistical representation of the presence and absence of each of the 20 instruments in the dataset. The large size of this dataset provided enough data to train the CNN models. It was noticed that the labels for each recording do not include every instrument present in the audio, however, the labels are the most prominent instruments in the audio. This was noticed when listening to the audio files and 'voice' was heard in some recordings but 'voice' was not a label for those audio files. Each audio file was transformed into a time-frequency representation and the representation was used as input to the CNN. CNNs learn features like lines and edges which can be used to identify images. While audio is not an image, the time-frequency representation of audio is 2D and can be treated like an image. A CNN treats the spectrograms like grayscale images to learn image-like features for classification. These spectrograms are the features that represent the audio. The spectrogram input type was a mel spectrogram in a dB scale. Mel spectrograms are similar to standard spectrograms, but mel spectrograms mimic human hearing by assigning high-frequency components to aggregated frequency bins.

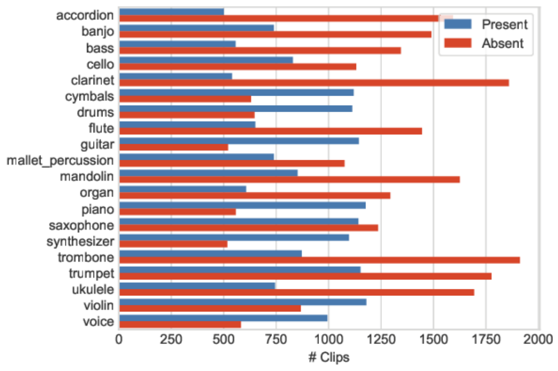


Fig. 1: Statistics of presence and absence of instruments in OpenMIC

B. Architectures

The project's approach was to transfer learn Resnet-50 and Vggish. ResNet-50 is an image classification network trained on the Imagenet dataset. While not designed to be used on audio, the network can be transfer learned to audio spectrograms. This means that feature extraction layers of the network can be used to look for lines and edges and the classifier layers can be used to learn the specific features of the dataset. The tunable parameters of the CNNs will be the convolutional layers and classification layers of the CNN. VGGish is a CNN architecture developed by Google Research. It is inspired by VGG, an image classification architecture. VGGish was trained on the AudioSet dataset, a corpus of 2 million 10 second audio files from YouTube. Unlike Resnet-50, the architecture was trained on audio spectrograms and was designed for audio classification. For these reasons, the transfer learning for VGGish was predicted to be more successful. VGGish converts CNN outputs into a 128D vector called an embedding layer. This embedding layer was predicted to make it more difficult to use gradient explainability methods. It was not expected that either architecture would beat the state-of-the-art models. The goal was to obtain sufficient accuracy such that explainability techniques could be used on the resulting networks. The architecture for both the ResNet-50 and VGGish are shown in Fig. 2 and Fig. 3 respectively.

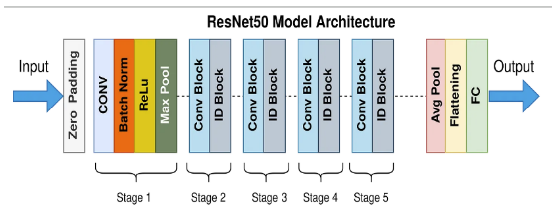


Fig. 2: ResNet 50 architecture

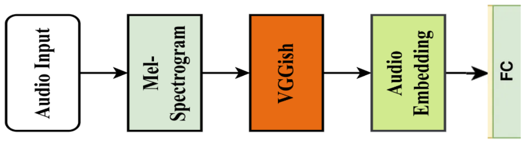


Fig. 3: ResNet 50 architecture

C. Explainability

After sufficient accuracy was determined, GradCAM and Deep Feature Factorization were used to explore the features of the input spectrograms that lead to the network's decisions. GradCAM uses the strength of gradients from a specified layer of a CNN to determine which pixels of an input image contribute most to any given target class. An example of how GradCAM can provide insight into what pixels of the images contribute to its decision-making is shown in Fig. 4. Using GradCAM can provide insight into what target classes a CNN performs well on and what target classes a CNN struggles with. In the context of instrument classification from mel spectrograms, GradCAM can reveal which frequencies and times on which the CNN most strongly bases its decision on. It can potentially reveal what time in the audio file an instrument is playing. Deep Feature Factorization (DFF) was also used to provide insight into model decision-making. DFF clusters every pixel in an input image into similar "concepts" or target classes. It reveals which target classes may be in the input image and it can be thought of conceptually as revealing what the network is thinking before it makes its final decision. An example of how DFF can show what the network is thinking before making its decision is shown in Fig. 6. In the context of instrument classification, this can reveal potential instruments that the network may have been considering before making its final decision.

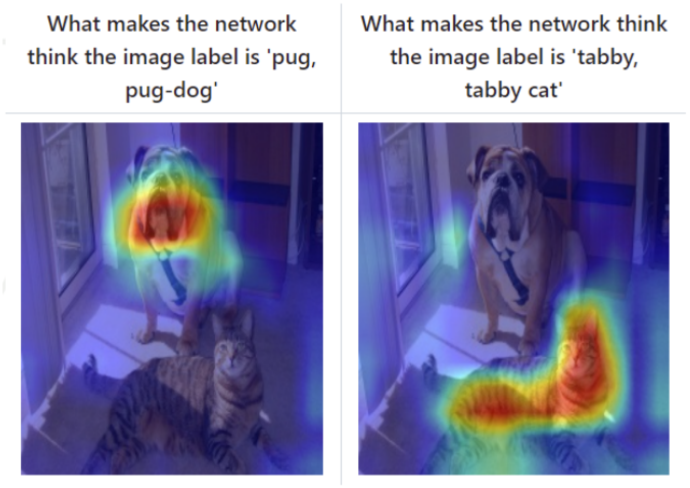

What makes the network think the image label is 'pug, pug-dog'
What makes the network think the image label is 'tabby, tabby cat'

Fig. 4: GradCAM example

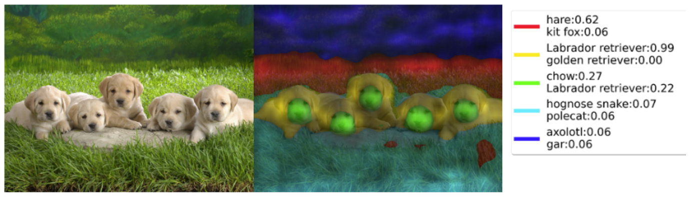


Fig. 5: Deep Feature Factorization example

IV. Experimental Results and Analysis

A. *Training GPU Resources*

Training CNNs can require high computational resources so, the limited computational ability needed to be considered. Without a GPU, training CNNs can take many hours or even days, however, Google and Kaggle provide free GPU resources in Python notebooks on their respective websites. Google Colab and Kaggle GPUs were used due to their limited but free GPU usage. Google Colab provides an unknown daily usage of GPU resources and little control over what type of GPU is used. Kaggle provides 30 hours of GPU usage per week and allows users to specify what type of GPU or TPU is used. With these GPU resources, the models can train in several hours as opposed to several days on CPU only. This meant all Python code was written in notebooks on Google Colab and Kaggle.

B. *Preprocessing*

In order to train the models, the .ogg audio files needed to be converted into spectrograms. This could have been done while the model trains, however, each time the model encounters an audio file, it would need to perform computations to generate the mel spectrogram. This motivated the conversion of each audio file into a mel spectrogram before training occurs. Instead of performing the mel spectrogram 'on the fly', the mel spectrograms are saved to disk for direct access. Each sample of the Openmic2018 dataset was converted into mel spectrograms using the Librosa library in Python. Librosa is an open-source audio processing library and has a built-in .ogg file reading function and a mel spectrogram function. The Librosa default values for the mel spectrogram were used. The spectrograms were saved as NumPy arrays in .npy files, a standard NumPy binary format. The trade-off of increased training speed was disk space because while the audio files in the dataset were 2.6 GB on disk, the spectrograms were about 4 GB on disk. In order to reduce the size of the spectrograms on the disk, the .npy files were saved as floats instead of doubles. While this sacrificed the accuracy of the spectrograms, it allowed for smaller disk requirements which were essential to training on Google Colab and Kaggle. The spectrograms generated by Librosa were used to train Resnet-50, however, different-sized spectrograms were required to train Vggish. This is because the implementation of Vggish used had a size requirement for the input spectrogram. Fortunately, the implementation of Vggish that was used has a function that generates the spectrograms that the model can accept.

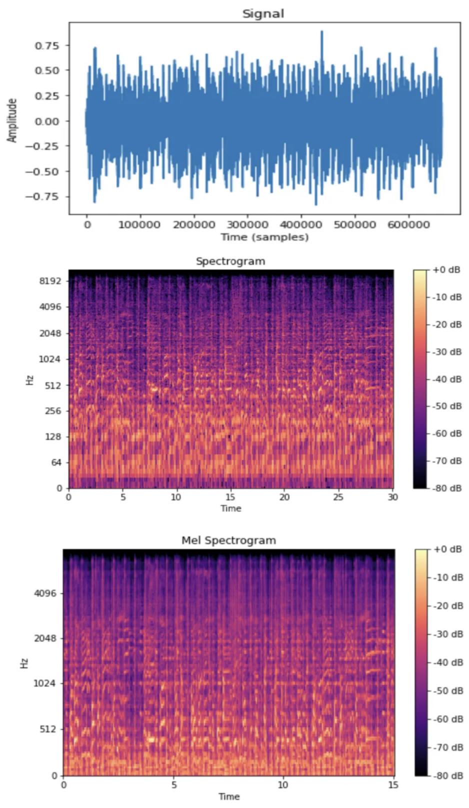


Fig. 6: Mel Spectrogram preprocessing

C. *Dataloader Development*

Dataloaders were developed for Resnet-50 and Vggish training. Dataloaders are a standard tool in Pytorch to train models because they allow for batch training.

D. *Model Development*

The CNN architectures were developed using Pytorch. Given the amount of data available in OpenMic2018, the CNN was able to be deep learn if necessary which validates the use of Resnet-50 and Vggish. The models were imported from Pytorch's built-in tools. Pytorch contains the pretrained Resnet-50 architecture and model weights due to its popularity. The fully connected layer of Resnet-50 was replaced to output 20 logits because there are 20 instrument classes in the dataset. A Vggish implementation was found on Pytorch's model hub which contains user-created models. The GitHub user 'harritaylor' created the Vggish model on Pytorch's model hub and provided pretrained weights for the model. The user's Pytorch implementation could also be found on Python's package index (pypi aka pip install) under the name 'torchvggish' with model weights provided by the user's GitHub page. A fully connected layer was added after the embedding layers of the network with a 20 logit output.

E. *Model Training*

Experimentation involved training more and more layers of each architecture, starting with training only a fully connected classification layer and ending with training the full network. For the Resnet-50 architecture, the model was first trained with only the fully connected layer unfrozen. Next, it was trained with the fully connected layer and the last convolutional block of Resnet-50 unfrozen. Finally, the whole network was trained without any frozen layers. For the Vggish architecture, the model was first trained with only the fully connected layer unfrozen. Next, it was trained with the fully connected layer and the 2 embedding layers unfrozen. Then, it was trained with the fully connected layer and all embedding layers unfrozen. Finally, the whole network was trained without any frozen layers. The models were trained with a batch size of 16 and were scheduled to train for 30 epochs unless stopped by early stopping. The batch size of 16 was chosen to balance training speed and stay within the RAM limit of the GPU resources. If the batch size was smaller, training was less efficient and took longer. If the batch size was larger, the GPU RAM limit would be reached and the runtime was shut down in Kaggle.

1) Data Split: The dataset was split randomly into training, testing, and validation sets using Pytorch's built-in functions. However, each data split was to be equivalent across the different layer training so that the networks were trained on the same data equally. The split percentages were 80% training, 15% testing, and 5% validation. The training set was used for training, the testing set was used for testing, and the validation set was used for early stopping.

2) Early Stopping: Early stopping is a method that prevents model overfitting. Once the model begins to overfit on the testing set, the validation set loss will increase. Therefore, during training, after every training epoch, the model was saved if the loss on the validation set had decreased. If the model's validation set loss increases for a specified number of epochs, known as patience, then the training stops. This way, the model with the lowest validation loss was saved. The patience used during training was 10.

3) Optimizer and Loss Function: The models were trained using an Adam optimizer, which is an extension of the stochastic-gradient descent optimization. The parameters were tuned to obtain the highest performance. A step reduction learning rate scheduler called 'StepLR' was used to optimize training step sizes. The gamma value used for the scheduler was 0.1. The starting learning rate used was 10^{-2} for the fully connected layer and 10^{-5} for any unfrozen non-fully connected layers. After 10 epochs, the learning rate decayed according to the learning rate scheduler. The Pytorch Adam optimizer also includes the option for L2 regularization to prevent overfitting. The L2 regularization weight decay was set to 10^{-4}. The loss function used was the binary cross entropy loss. This loss function is used in binary classification tasks but is also commonly used in multilabel classification tasks because it maps logits into values between 0 and 1, which represent class probabilities. In Pytorch the loss function is called 'BCEWithLogitsLoss'.

F. Training Analysis

After each epoch of training, the training loss, the validation loss, the validation macro F1, the validation micro F1 score, and the validation Jaccard score were displayed so that training could be monitored. Training occurred intermittently over the course of 4 weeks mostly using Kaggle GPU resources. The results of training Resnet-50 and Vggish are summarized in TABLE I and TABLE II, respectively.

Layers Trained	# of Param-eters	Validation Macro F1	Validation Micro F1	Validation Jaccard Score
Fully Connected (FC) Only	~41k	0.2861	0.2840	0.1773
FC + last conv. block	~15m	0.4360	0.4437	0.3420
Full Network	~24m	0.5469	0.5526	0.4477

TABLE I: Resnet-50 Training Results Table

Layers Trained	# of Param-eters	Validation Macro F1	Validation Micro F1	Validation Jaccard Score
Fully Connected (FC) Only	~25k	0.5539	0.5504	0.4215
FC + 2 embed-ding layers	~17m	0.6091	0.6060	0.4891
FC + all embed-ding layers	~67m	0.6424	0.6394	0.5395
Full Network	~72m	0.6465	0.6423	0.5473

TABLE II: Vggish Training Results Table

G. Testing Analysis

After training the models, the results obtained from ResNet 50 and VGGish were evaluated. Evaluation metrics included macro F1 score, micro F1 score, and Jaccard score. These metrics provide high-level overviews for multilabel classification tasks. They are different ways to measure the aggregate performance of the true positives, false positives, and false negatives that the network classifies. This is especially important in a multilabel classification task because each audio file can be labeled with multiple instruments so each label can have its own F1 score. With 20 instrument classes, there would be 20 different F1 scores, so aggregation can simplify the evaluation to a single F1 score. The results of testing Resnet-50 and Vggish on the testing set are summarized in TABLE III and TABLE IV, respectively. The results showed that the model performed relatively well considering that state-of-the-art models achieve a macro F1 score of about 0.8 [9] [10]. Again, the goal of the project was not to beat the state of the art. For the purposes of explainability, the results are sufficient. In comparison to each other, Vggish performed better than Resnet-50 in every evaluation metric and at every training level. This is likely because Vggish was designed for audio classification and therefore trained on spectrograms. Resnet-50 needed to learn the spectrogram structures because Resnet-50 was pre-trained on natural images, which differ from spectrograms significantly. Furthermore, the full Vggish architecture contains more model parameters than Resnet-50 which would warrant higher performance.

Layers Trained	# of Parameters	Validation Macro F1	Validation Micro F1	Validation Jaccard Score
Fully Connected (FC) Only	∼41k	0.2891	0.2913	0.1843
FC + last conv. block	∼15m	0.4804	0.4242	0.2973
Full Network	∼24m	0.5433	0.5461	0.4379

TABLE III: Resnet-50 Testing Results Table

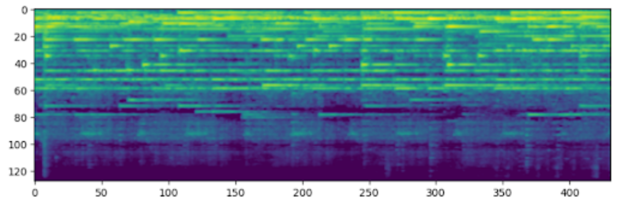
Layers Trained	# of Parameters	Validation Macro F1	Validation Micro F1	Validation Jaccard Score
Fully Connected (FC) Only	∼25k	0.5665	0.5673	0.4372
FC + 2 embedding layers	∼17m	0.6104	0.6118	0.4978
FC + all embedding layers	∼67m	0.6382	0.6384	0.5381
Full Network	∼72m	0.6405	0.6407	0.5443

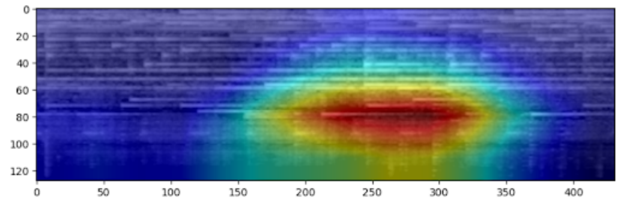
TABLE IV: Vggish Testing Results Table

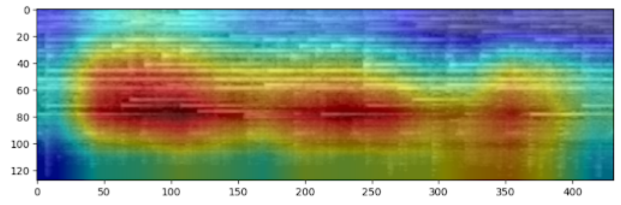
H. Explainability

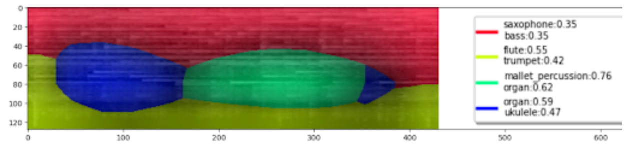
After sufficient training and validation, a Pytorch visualization library was created by GitHub user 'jacobgil' and was used to implement GradCAM and DFF. This provided insight into which types of instruments the networks accurately classify and struggles to classify. The library creates a heatmap that overlays on the mel spectrogram input. Visualizing each of the 20,000 audio file classifications in the dataset would have been too time-consuming, so visualization was performed on only some of the audio files.

An example of using GradCAM on a Resnet-50 decision can be seen in Fig. 7. The model predicts the audio contains mallet percussion and organ. The GradCAM explanation shows that there are frequencies in the middle of the audio that tell the network that the audio contains mallet percussion. It also reveals that the network believes that there is an organ played throughout the audio. However, the ground truth is mallet percussion. There are spectral features in the spectrogram that the network believes to be characteristic of an organ. The DFF of the model decision reveals that the network also noticed concepts of saxophone, bass, trumpet, or flute within the spectrogram as well. There are potential applications for using the DFF to label audio when making or improving datasets. If the performance of the model was high enough, the DFF could potentially be used to identify instruments in the audio that the human labelers missed when creating the dataset.


(a) Original Mel Spectrogram of sample


(b) GradCAM overlay of that explains why mallet percussion

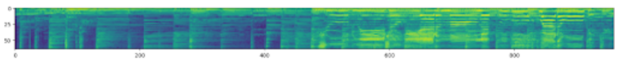

(c) GradCAM overlay of why organ



(d) GradCAM overlay of why organ

Fig. 7: (a - c) Example of GradCAM visual explanation on a Resnet-50 classification. The predicted instruments were mallet percussion and organ. The ground truth instrument was mallet percussion. (d) Example of DFF visual explanation on a Resnet-50 classification.

An example of using GradCAM on a Vggish decision can be seen in Fig. 8. The model predicts the audio contains a saxophone. The GradCAM explanation shows that there are frequencies sporadically throughout the spectrogram that tell the network that the audio contains a saxophone. The explanation with GradCAM is less revealing with the Vggish architecture. Furthermore, the DFF was unable to be performed. One explanation for the inability to visualize Vggish decisions is the way the spectrograms were created using the Vggish implementation and the way the spectrograms flow through the network. The spectrograms were created by splitting the original audio into 0.96 second windows. Then individual spectrograms were created from each window and concatenated together. The implementation of the Vggish model could only take a single window as input at a time, so a workaround was developed by treating each window as a separate input to the model and then concatenating them together. This is what causes the GradCAM overlay to look discontinuous. This workaround makes training faster because each window is treated as a separate 'batch' from the model's point of view, but it makes it difficult to visualize network decisions. Another issue with using Vggish to visualize network decisions is the way the embedding layers reshape the output of the convolutional layers. Reshaping appears to be the issue with poor network visual explanations. Future work could be developing a Vggish implementation that allows for 10-second input to the network, however, note that training would take much longer because convolutional layers would need to learn over larger input spectrograms and Vggish may not perform as well because the pretrained model weights are designed for a specific input spectrogram size. A custom network may perform well and still provide insightful visual explanations.


(a) Original Mel Spectrogram of sample

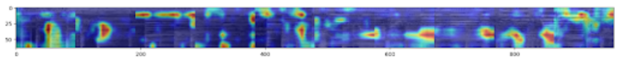

(b) GradCAM overlay of that explains why mallet percussion

Fig. 8: Example of GradCAM visual explanation on a Vggish classification. The predicted instrument was the saxophone. The ground truth instruments were guitar and saxophone.

V. Conclusion

Overall, the project demonstrates the use of image-processing techniques in audio and speech processing. The transfer learning of Resnet-50 and Vggish perform relatively well. They do not achieve state-of-the-art performance, but this was not expected. Furthermore, multilabel, multiphonic classification is considered difficult and often requires more advanced networks. An advantage of less advanced CNN models is that they can provide visual explanations. The use of GradCAM and DFF provides insight into the networks which are often considered 'black boxes'. Concepts from adjacent signal processing disciplines like image processing can be used in audio processing. While the project used 2D CNNs and audio spectrograms, 1D CNNs can be used on raw audio data. With 1D convolutions, 1D GradCAM can be used. The project also demonstrates potential issues with visual explanations. If reshaping occurs within a network, it may be more difficult to generate insightful visual explanations. Vggish performed better than Resnet-50, but Resnet-50 provided better visual explanations than Vggish. Resnet-50 provided insightful Grad-CAM overlays and DFFs. Vggish GradCAM overlays were less insightful, and DFF was not able to be performed on Vggish.

Future work includes using more advanced network models such as recurrent neural networks (RNNs) and transformer models. Hybrid networks can be used like CRNNs and CNN-RNNs. These networks include elements of CNNs and RNNs and have been used in audio classification in literature. The RNN and transformer aspects of more advanced networks may require different explainability techniques that are not visual. Future work can be performed on the Vggish model. The Vggish model can be designed to allow any spectrogram input size and can be designed without embedding layers. This may allow for better GradCAM overlays and allow for DFF to be performed. However, this may no longer be considered transfer learning because the pretrained weights of Vggish might become useless. The architecture might require training from scratch in this case. A custom model may also be trained from scratch. This has been done in literature by university students in literature. A custom network that uses few reshapings and ReLU activation functions between layers would allow for visual explanations like GradCAM and DFF to be performed.

VI. Code

The code for this project is available on GitHub

References

[1] K. Choi, G. Fazekas, M. Sandler, and K. Cho, ”Convolutional recurrent neural networks for music classification,” 2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), New Orleans, LA, USA, 2017, pp. 2392-2396, doi: 10.1109/ICASSP.2017.7952585.

[2] K. Avramidis, A. Kratimenos, C. Garoufis, A. Zlatintsi and P. Maragos, ”Deep Convolutional and Recurrent Networks for Polyphonic Instrument Classification from Monophonic Raw Audio Waveforms,” ICASSP 2021 - 2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), Toronto, ON, Canada, 2021, pp. 3010-3014, doi: 10.1109/ICASSP39728.2021.9413479.

[3] Qu, Y., Li, X., Qin, Z., & Lu, Q. (2022). Acoustic scene classification based on three-dimensional multi-channel feature-correlated deep learning networks. Scientific reports, 12(1), 13730. https://doi.org/10.1038/s41598-022-17863-z

[4] Choi, K., Fazekas, G., & Sandler, M. (2016). Explaining deep convolutional neural networks on music classification. arXiv preprint arXiv:1607.02444.

[5] Rojat, T., Puget, R., Filliat, D., Del Ser, J., Gelin, R., & Díaz-Rodríguez, N. (2021). Explainable artificial intelligence (xai) on timeseries data: A survey. arXiv preprint arXiv:2104.00950.

[6] R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh and D. Batra, ”Grad-CAM: Visual Explanations from Deep Networks via Gradient-Based Localization,” 2017 IEEE International Conference on Computer Vision (ICCV), Venice, Italy, 2017, pp. 618-626, doi: 10.1109/ICCV.2017.74.

[7] Collins, E., Achanta, R., & Susstrunk, S. (2018). Deep feature factorization for concept discovery. In Proceedings of the European Conference on Computer Vision (ECCV) (pp. 336-352).

[8] Humphrey, Eric J., Durand, Simon, & McFee, Brian. (2018). OpenMIC-2018 (v1.0.0) [Data set]. Zenodo. https://doi.org/10.5281/zenodo.1432913

[9] Zhong, Z., Hirano, M., Shimada, K., Tateishi, K., Takahashi, S., & Mitsufuji, Y. (2023). An Attention-based Approach to Hierarchical Multi-label Music Instrument Classification. arXiv preprint arXiv:2302.08136.

[10] Gururani, S. K. (2020). Weakly Supervised Learning for Musical Instrument Classification (Doctoral dissertation, Georgia Institute of Technology).