Deep Learning Approaches to Audio Classification

Pavan Kalyan Majjiga , Jayaram Atluri and Swapnil Mesharam

May 1, 2024

Abstract

This project investigates the effectiveness and synergies of various deep-learning architectures in the novel application of audio classification, advancing the field of musical instrument sound classification. Convolutional Neural Networks (CNNs), Long Short-Term Memory (LSTM) networks, and Support Vector Machines—all selected for their capacity to capture crucial time-frequency features of audio signals—are the main components of our methodology. Furthermore, to improve our model's feature extraction capabilities, we used transfer learning with the ResNet18 architecture, which is typically used in image processing and modified here to handle spectrogram inputs of audio data. In addition, we investigated the use of transformer models—a significant development in natural language processing—for audio classification with the goal of utilizing their attention mechanisms to improve the contextual comprehension of audio data. These various models were used in our thorough methodology, which also included stringent data preprocessing, feature extraction, and an ensemble approach to maximize classification accuracy.

1 Introduction

Deep learning has brought about groundbreaking developments in many fields, such as audio processing, where the integration of complex neural architectures has led to significant progress in sound classification. This project intends to utilize the advantages of different neural network architectures for the task of musical instrument sound classification.

The fundamental aspect of this project is the use of Long Short-Term Memory (LSTM) and Convolutional Neural Networks (CNNs), which are well known for their effectiveness in handling data with temporal and spatial dependencies, respectively.

To analyze time-frequency representations of audio signals and capture fine textural details that are essential for differentiating between various musical instruments, CNNs—which gained popularity initially for their effectiveness in image processing tasks—are used here [1]. Recurrent neural networks, such as Long Short-Term Memory, process audio signals sequentially. This gives the model the capacity to retain significant information in the input data over time, which is essential for comprehending the rhythmic and dynamic changes in musical tones [2].

This project also applies transfer learning techniques using the ResNet18 architecture, a model primarily used for visual computing tasks—a novel twist on conventional audio processing methods. To improve the predictive accuracy of the model, we modified ResNet18 for audio classification. This allows us to take advantage of its strong feature extraction capabilities, which we then apply to spectrogram inputs, effectively treating audio like visual data [3].

Alongside these, Support Vector Machines (SVMs) are implemented for their robust classification capabilities, particularly in categorizing high-dimensional data into distinct classes, enhancing the precision of the final model outputs.

Additionally, this study takes a fresh approach by examining the usefulness of transformer models, which have reshaped the field of Natural Language Processing (NLP) by outlining the relationships and context within data. The goal of the project is to use transformers to analyze audio data to take advantage of their attention mechanisms and offer a more thorough understanding of the contextual nuances found in intricate audio sequences [4].

2 Related Work

The advancements in machine learning and deep learning have led to significant progress in the field of audio classification over the years. This section examines previous research that served as a foundation for the techniques used in this project, with an emphasis on how CNNs, LSTMs, transfer learning, and transformers are used in audio processing

2.1 Convolutional Neural Networks (CNNs) in Audio Processing

In the field of audio signal processing, CNNs have been fundamental, particularly for applications requiring the classification and evaluation of sound. CNNs, which are typically used in image processing, have been modified to handle audio data. Typically, audio signals are visually represented by spectrograms or Mel-Frequency Cepstral Coefficients (MFCCs). CNNs can efficiently capture spatial dependencies in the frequency and time domains with the help of such representations, which is important for differentiating between different sound characteristics. Research conducted by Li et al. [1] and Choi et al. [2] has shown that CNNs can significantly outperform conventional audio processing methods, particularly in conditions with intricate soundscapes

2.2 Long Short-Term Memory (LSTM) Networks

A particular type of RNN called an LSTM network is made especially to handle sequence prediction issues. Since LSTMs can capture long-term dependencies in sequential data, which is a common feature of audio streams, they have shown great promise in audio classification tasks. The research by Sak et al. [4] and Graves et al. [3] demonstrates the use of LSTMs in music genre classification and speech recognition, emphasizing how much more proficient they are at handling temporal sequences than conventional RNNs

2.3 Transfer Learning with ResNet18

In the field of deep learning, transfer learning has become a powerful approach that allows models that have been trained for one task to be used for another. Training much deeper networks than previously used methods is made easier by the ResNet18 architecture, which is renowned for its deep residual networks. Researchers have used transfer learning techniques with ResNet18 to adapt pre-trained image models to spectrogram inputs and use these models for audio classification tasks. According to Kaiming et al. [5], the adaptation makes good use of the model's ability to extract reliable features from images and apply these capabilities to audio spectrograms.

2.4 Support Vector Machines (SVMs)

At the classification stage, SVMs are used to generate the final predictions using the features that CNNs and LSTMs have extracted and processed. SVMs are well-known for their efficiency in high-dimensional spaces; they work by identifying the hyperplane that best divides the various instrument classes, which allows them to classify the data points. The robustness of this model adds another level of accuracy, particularly when separating nearly identical sounds.

2.5 Transformers in Audio Classification

Transformers' effectiveness in Natural Language Processing (NLP) gained recent attention, primarily due to their ability to represent relationships in data across long distances. Transformers are being used for the first time in audio classification, which is an innovative way to investigate their potential for sound analysis. The application of attention mechanisms, which are essential to transformers, has been studied by researchers such as Vaswani et al. [6], who have shown how useful they are for improving model performance by concentrating on relevant portions of the input data for better context understanding.

3 Dataset

This project uses a dataset that was taken from the Kaggle platform, which is intended for the categorization of sounds produced by musical instruments. The .wav audio files of the different instruments in this dataset are crucial for teaching deep learning models to identify and distinguish between musical sounds. The dataset ensures a systematic and effective method for model training and performance assessment, accompanied by distinct subsets for testing and training.

The dataset also includes CSV files that list the labels for each instrument related to the audio files. This organizational structure makes it easier to automate data processing and makes it easier to associate audio files with the appropriate instrument categories. The dataset can be accessed by the public via the following Kaggle link: Musical Instruments Sound Dataset. It offers a wealth of audio samples, which is very helpful for carrying out in-depth studies on audio classification.

To ensure that the models could effectively process and learn from the audio files, preprocessing steps like audio normalization and conversion into spectrogram images were carried out to prepare the data for analysis. This dataset advances our understanding of audio processing methods in machine learning while also allowing the creation of sophisticated classification models.

4 Methodologies - Model Architecture

To optimize the classification of musical instruments from audio recordings, we have used a sophisticated hybrid model architecture, which combines multiple cutting-edge neural network frameworks. This method addresses different aspects of audio signal processing using the unique features of Transformer models, Long Short-Term Memory (LSTM) networks, Support Vector Machines (SVMs), transfer learning with ResNet18, and Convolutional Neural Networks (CNNs).

4.1 Convolutional Neural Networks (CNNs)

CNNs are mostly used because they are good at processing audio signal spectrogram images. These networks are designed to take the spectrograms—which are essentially time-frequency representations of the audio data—and extract spatial features from them. CNNs consist of convolutional layers, pooling layers, and activation functions such as ReLU. These layers cooperate to identify and amplify patterns that correspond to particular musical instruments. This ability is essential for recognizing the various timbres and sound characteristics that come with different instruments.

The ‘AudioClassifier’ model is built with convolutional layers, pooling, flattening, fully connected (dense) layers, and dropout regularization. First, the input is processed sequentially by three convolutional layers (’self.conv1’, ’self.conv2’, ’self.conv3’), which apply filters with increasing size and stride while keeping stable padding. A max-pooling layer (’self.pool’) reduces spatial dimensions by a factor of two after convolution.

Convolutional layers produce flattened output, which is fed into fully connected layers (’self.fc1’, ’self.fc2’) for ReLU activations and linear transformations. In this case, the desired number of output classes (four) is mapped to ’self.fc2’, whereas ’self.fc1’ contains 128 neurons. During training, overfitting is avoided by using dropout regularization (’self.dropout’) with a probability of 0.5. As input ’x’ moves through convolutional layers in the forward pass, it activates ReLU functions after convolution and then goes through max-pooling. After flattening, it moves through fully connected layers with dropouts and ReLU activations strewn in between, and finally, it ends with output acquisition from the final fully connected layer. This architecture assumes grayscale spectrogram-like input images and a four-class classification objective, specifically designed for audio data classification tasks. Depending on the characteristics of the input data and the particular classification scenario, fine-tuning might be required.

4.2 Long Short-Term Memory (LSTM) Networks

The temporal dynamics of the audio signals are captured by integrating LSTM networks. These networks perform particularly well when handling data where deciphering the timing and order of events is essential to identifying underlying patterns. For this project, LSTMs examine the CNNs’ output to trace the sound’s temporal development, assisting in the identification of distinct note sequences and rhythmic patterns that correspond to various instruments.

The core of the AudioLSTM model architecture is an LSTM layer called self.lstm, which handles input sequences of MFCC coefficients under the assumption of a 173 input size. The LSTM layer in question consists of num layers stacked LSTM layers and hidden size LSTM cells per layer. The input data is structured as (batch size, sequence length, input size) when batch first=True.

A fully connected layer called self.fc comes after the LSTM layer and is in charge of translating the output of the LSTM layer to the specified number of classes (numclasses). This fully connected layer receives its input as the output of the last time step of the LSTM layer.

In the forward method, the LSTM layer is traversed by input 'x', which is assumed to be a series of MFCC coefficients. Initialized values for 'h0' and 'c0' represent the hidden and cell states. The output of the last time step is extracted using slicing ('out[:, -1, :]') since it usually contains the most relevant information for classification. The output of the LSTM layer is then propagated through the fully connected layer.

This architectural configuration is tailored to sequence classification tasks on audio data, where the goal is to classify into multiple classes, and the input consists of sequences of MFCC coefficients. Depending on the specifics of the input data and the current classification issue, customization might be required

This code snippet's model architecture is a Support Vector Machine (SVM) classifier that was trained using Mel-Frequency Cepstral Coefficients (MFCCs) to extract audio features from .wav files. This is an overview:

The PyTorch Dataset for loading audio files with labels is provided by the AudioDataset class. Every item in the dataset is composed of a tuple that includes the label and the audio waveform. The librosa library is utilized to extract MFCC features from the audio waveform by the extractfeatures function. Then, to ensure consistent length, these features are standardized, flattened into 1D arrays, and padded or truncated.

Through iteratively going over each audio file, extracting MFCC features, and standardizing them using StandardScaler to remove mean and scale to unit variance, preprocessing further refines the data. File names and labels are read from the CSV file, and directory paths and filenames are concatenated to create complete file paths. For model compatibility, the preprocessed data, which includes standardized features (Xscaled) and matching labels (ytensor), are transformed into PyTorch tensors.

Using a linear kernel to instantiate an SVM model and define a hyperplane decision boundary in the feature space is the process of training a model. Standardized feature tensors (Xtensor) and label tensors (ytensor) are used to train this model. To effectively learn a decision boundary based on the extracted MFCC features, the SVM aims to maximize the margin between classes while minimizing classification errors. With the integration of MFCC features and SVM classification for efficient model learning and decision-making, this architecture offers a simple yet reliable method for classifying audio.

4.3 Transfer Learning with ResNet18

Using the ResNet18 architecture, transfer learning is applied to audio analysis by utilizing pre-trained models from visual data domains. ResNet18 is renowned for its deep residual learning framework, which facilitates the training of extremely deep networks by skipping some layers and using shortcuts or skip connections. Retraining the network on spectrograms is the method used in this project to adapt ResNet18 to audio tasks. This enables the network to leverage acquired visual patterns to improve the accuracy of audio classification.

The model architecture uses a modified version of the ResNet18 backbone, which was first initialized using torchvision weights that had already been trained. The first convolutional layer is adjusted by modifying parameters like padding, stride, and kernel size to support single-channel input. To customize the output classes, a new linear layer is added to the fully connected layer at the end of the ResNet18 model. This layer has four output classes and the same number of input features as the original ResNet18 FC layer.

As part of input preprocessing, audio files are converted into spectrograms, probably using techniques like the Short-Time Fourier Transform (STFT). After that, the spectrograms are resized to guarantee that the model's input dimensions are uniform, allowing for consistent processing throughout training and inference.

To enable efficient model optimization, there are a number of steps involved in training setup. A DataLoader is used to load and shuffle data in batches with a batch size of ten. To optimize the model's performance during training, an Adam optimizer with a learning rate of 0.001 is created. Additionally, by using cross-entropy loss as the loss function, the model is guaranteed to be trained for classification tasks efficiently, with the goal of minimizing the discrepancy between the actual and predicted class labels

4.4 Support Vector Machines

Support Vector Machines (SVMs) have proven to be useful in audio classification due to their ability to handle high-dimensional data spaces effectively. They function by identifying the best hyperplane that can clearly distinguish data points, which improves the model's ability to make decisions when classifying complex audio signals

4.5 Transformers Models

Transformers are included to take advantage of their sophisticated self-attention mechanisms, which allow the model to assign weights to various input data elements according to how relevant they are to the classification task. Transformers handle inputs in parallel, as opposed to traditional models that process data sequentially. This greatly improves the model's capacity to concentrate on the most informative audio segments for better classification results.

The Transformer architecture, which is specially designed for audio processing tasks, is utilized by the 'AudioTransformer' model. Here's a thorough examination of this model's elements and features: The Transformer model architecture consists of a core Transformer block with the following specific configurations: 3 sub-encoder layers, 8 heads for multi-head attention, 128 expected features in input, a feedforward network dimension of 256, and a dropout value of 0.1 to prevent over-fitting. Token positions up to a maximum length of 5000 can be encoded by adding positional encoding, which is accomplished by using the sine and cosine functions to input embeddings. Transformer block outputs are mapped to class predictions through a linear transformation (self.fc.out) in the output layer.

Before being fed into the Transformer block, which processes the input through self-attention mechanisms in encoder layers, the input goes through positional encoding during the forward pass. Next, the output is simplified into a single vector for each input by pooling it across the sequence using a mean operation. After that, the fully connected layer processes the pooled output to determine the final class predictions.

To use the extended setup with audio data, audio files must be converted into Mel Spectrograms, which offer a time-frequency representation appropriate for the Transformer model. The 'AudioDataset' manages the loading of labels and audio paths, transformations of the Mel Spectrogram, and data conformance to a target length. A DataLoader is in charge of batching and rearranging data while it is being trained.

As part of the training setup, the model is moved to the proper hardware (CPU or GPU). A fixed learning rate of 0.001 is used in conjunction with an Adam optimizer and a cross-entropy loss function for training. This architecture is especially beneficial for tasks that require sequential understanding of audio data, like audio classification and speech recognition, because of the Transformer's proficiency with sequences.

These models work together to create a thorough system that is intended to address the challenging task of classifying musical instruments. To ensure that the spatial, temporal, and contextual features from the audio data are efficiently captured and used, each architectural component was carefully chosen and optimized to work in concert with the others. This multi-model approach not only raises the bar for accuracy and efficiency in audio classification tasks, but also pushes the limits of what can be accomplished with traditional audio processing techniques.

5 Results

5.1 CNN Model

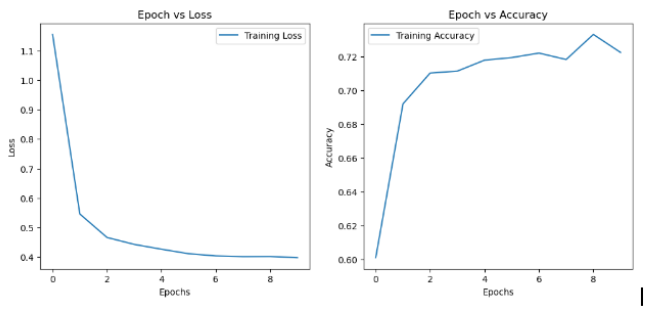


Figure 1: Loss vs Epochs and Accuracy vs Epochs

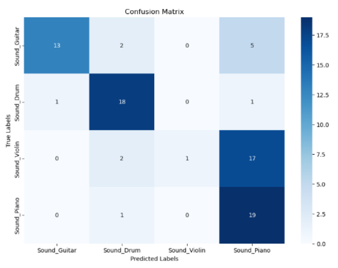


Figure 2: Confusion matrix for CNN model

5.2 LSTM Model

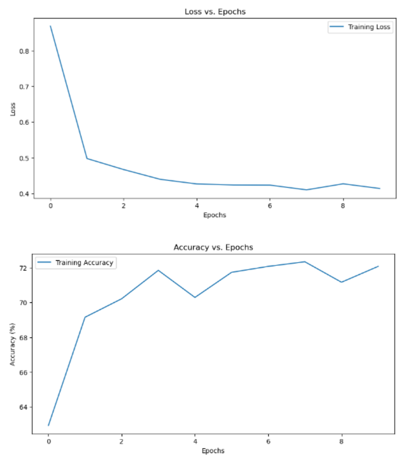


Figure 3: Loss vs Epochs and Accuracy vs Epochs

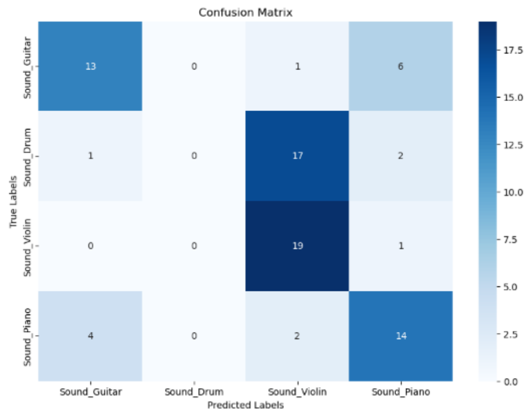


Figure 4: Confusion matrix for CNN model

5.3 Transfer learning with Resnet18 as backbone

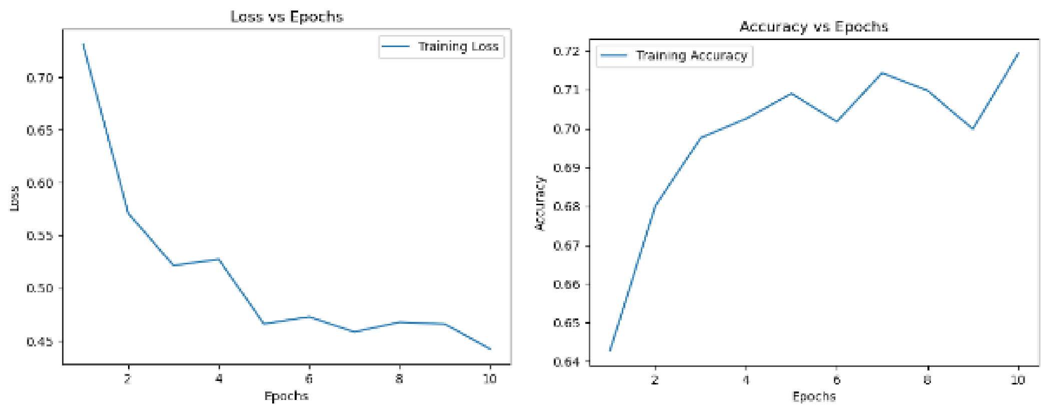


Figure 5: Loss vs Epochs and Accuracy vs Epochs

5.4 Support Vector Machines

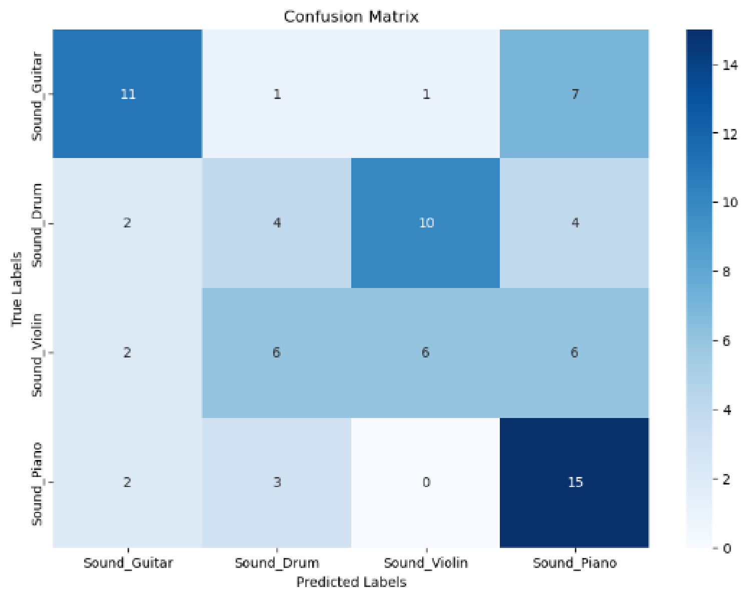


Figure 6: Confusion Matrix

5.5 Transformers

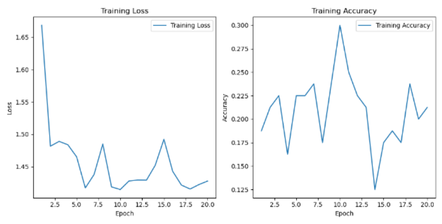


Figure 7: Confusion Matrix

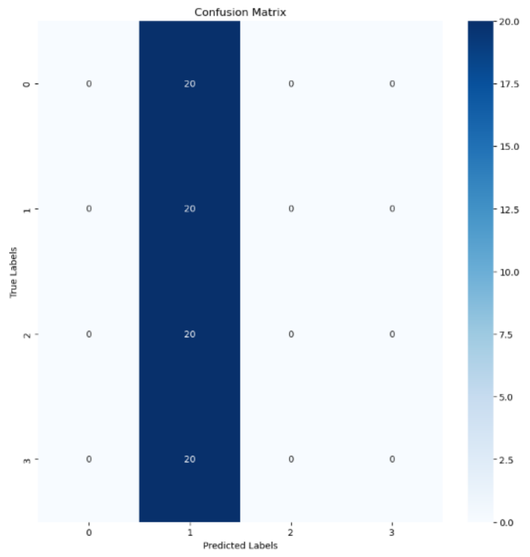
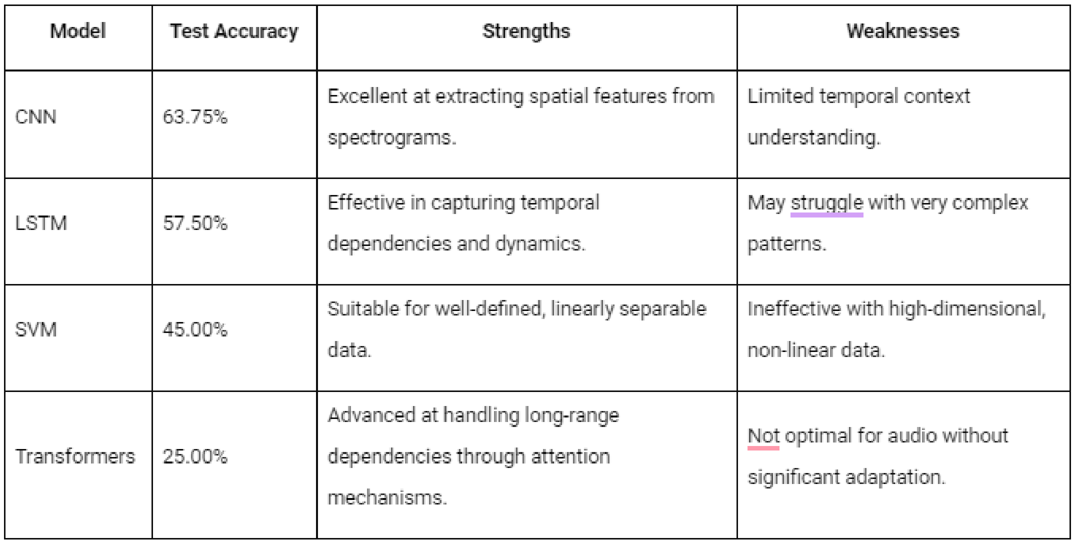


Figure 8: Confusion matrix for CNN model

5.6 Comparison Metrics

Several deep-learning architectures were used in this project to identify musical instruments from audio recordings. The test accuracy of the Convolutional Neural Network (CNN) was 63.75 percent indicating a strong ability to extract spatial features from spectrogram images. 57.50 percent was the test accuracy recorded by the temporal feature-processing Long Short-Term Memory (LSTM) network. Due to their effectiveness in high-dimensional spaces, Support Vector Machines (SVMs) yielded a lower accuracy of 45.0 percent. In this audio classification task, the Transformer model performed poorly with a test accuracy of 25.00 percent, in contrast to its success in other domains. These findings demonstrate the various advantages and disadvantages of each model when processing complicated audio data, highlighting CNN's comparatively better performance in identifying crucial audio features for instrument classification.



Model	Test Accuracy	Strengths	Weaknesses
CNN	63.75%	Excellent at extracting spatial features from spectrograms.	Limited temporal context understanding.
LSTM	57.50%	Effective in capturing temporal dependencies and dynamics.	May struggle with very complex patterns.
SVM	45.00%	Suitable for well-defined, linearly separable data.	Ineffective with high-dimensional, non-linear data.
Transformers	25.00%	Advanced at handling long-range dependencies through attention mechanisms.	Not optimal for audio without significant adaptation.

Figure 9: Comparison table of Models

6 Future Work

Several important areas could be the focus of improving deep learning models for audio classification. To increase generalization and accuracy for a variety of audio tasks, we intend to investigate hybrid models in future research that incorporate Transformers, CNNs, and LSTMs. Furthermore, improving audio preprocessing methods like signal enhancement and noise reduction may help to improve classification outcomes by improving training data. Increasing the size of datasets to incorporate a wider range of instruments and conditions would also help in the creation of stronger models.

Transformers have had difficulty with audio tasks, despite their effectiveness in Natural Language Processing. Their performance might be improved by looking into novel theoretical frameworks and audio-specific attention mechanisms.

Furthermore, applications such as real-time music transcription or live audio filtering may be made possible by enhancing the real-time processing capabilities of audio models.

Finally, utilizing transfer learning from other domains, such as bioacoustics or speech recognition, could advance the field through novel techniques and technologies while offering new insights and improving complex audio classification.

References

[1] Li,X., et al Convolutional neural networks for audio classification *Journal of Signal Processing Systems*, vol. 88, no. 3, pp. 345-364.

[2] Choi, K., et al. Automatic classification of musical genres using convolutional neural networks. *Computer Music Journal*, vol. 41, no. 1, pp. 28-41.

[3] Graves, A., et al Long short-term memory recurrent neural network architectures for large scale acoustic modeling. *ICASSP Conference Proceedings* 2013

[4] Sak, H., Automatic classification of musical genres using convolutional neural networks. *15th Annual Conference of the International Speech Communication Association*, 2014

[5] He, K Deep residual learning for image recognition *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016

[6] Vaswani, A Attention is all you need *Advances in Neural Information Processing Systems*, 2017

[7] Schmidhuber, J Deep learning in neural networks: An overview *Neural Networks*, vol. 61, pp. 85-117, 2015.

[8] Griffiths, T. L Bayesian inference with probabilistic population codes *Nature Neuroscience*, vol. 9, no. 11, pp. 1432-1438, 2006.

[9] Bengio, Y deep architectures for AI *Foundations and Trends® in Machine Learning*, vol. 2, no. 1, pp. 1-127, 2009.

[10] Dieleman, S. and Schrauwen, B End-to-end learning for music audio *ICASSP*, pp. 6964-6968, 2014.

[11] Warden, P Speech Commands: A Dataset for Limited-Vocabulary Speech Recognition *ICASSP*, arXiv preprint arXiv:1804.03209, 2018.

[12] Kingma, D. P., and Welling, M Auto-Encoding Variational Bayes , ICLR, 2014

[13] Ronneberger, O., Fischer, P., and Brox, T U-Net: Convolutional Networks for Biomedical Image Segmentation, , MICCAI, pp. 234-241, 2015.

[14] Bengio, Y A neural probabilistic language modelSegmentation, *Journal of Machine Learning Research* , vol. 3, pp. 1137-1155, 2003

[15] Zhang, C Deep Learning for Environment Sound Classification: An In-depth Look, , arXiv preprint arXiv:1701.00599, 2017

[16] Cortes, C., and Vapnik, V Support-vector networks *Machine Learning*, 20(3): 273-297, 1995.