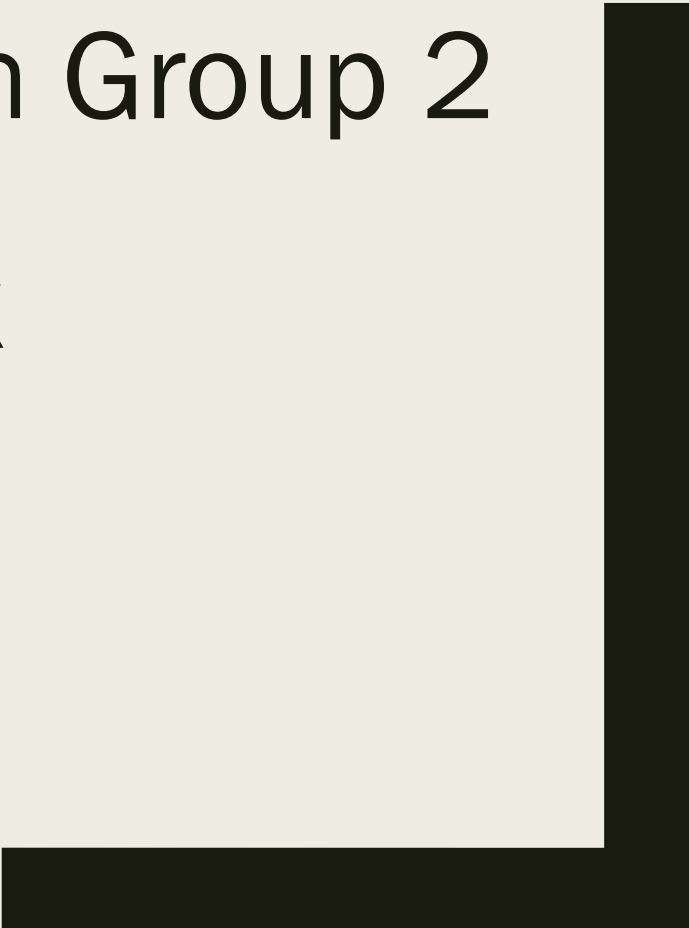




# Final Project Presentation Group 2

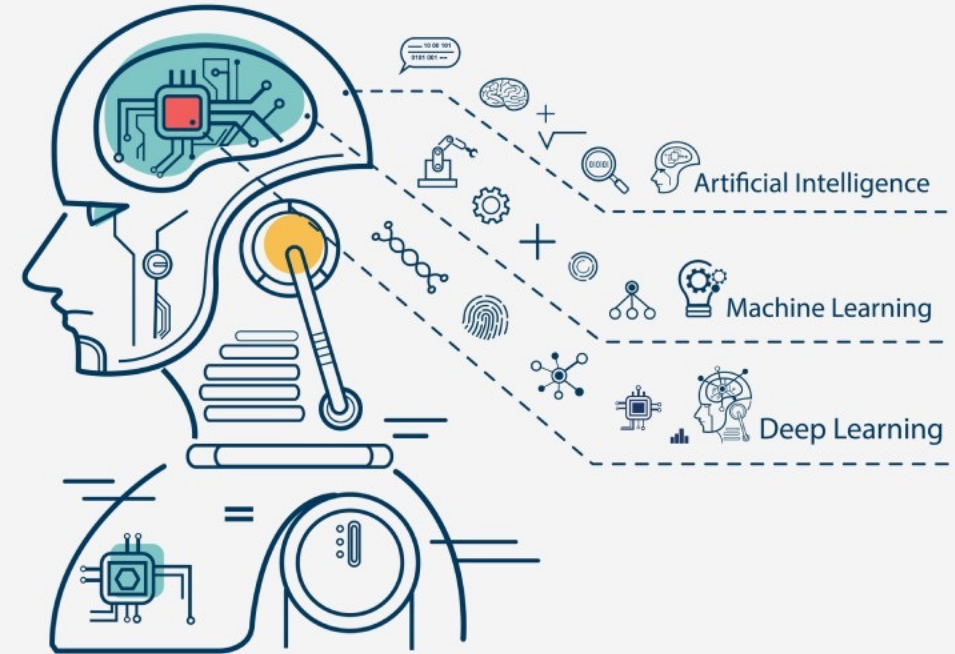
## UrbanSounds8k

By: Tristin Johnson  
DATS 6203 – Machine Learning II  
December 6<sup>th</sup>, 2021



# Table of Contents

- Introduction
- Data Analysis
- Data Preprocessing
- CNN Architecture
- Training & Validation
- Results - Testing the Model
- Future Work
- Conclusion
- References



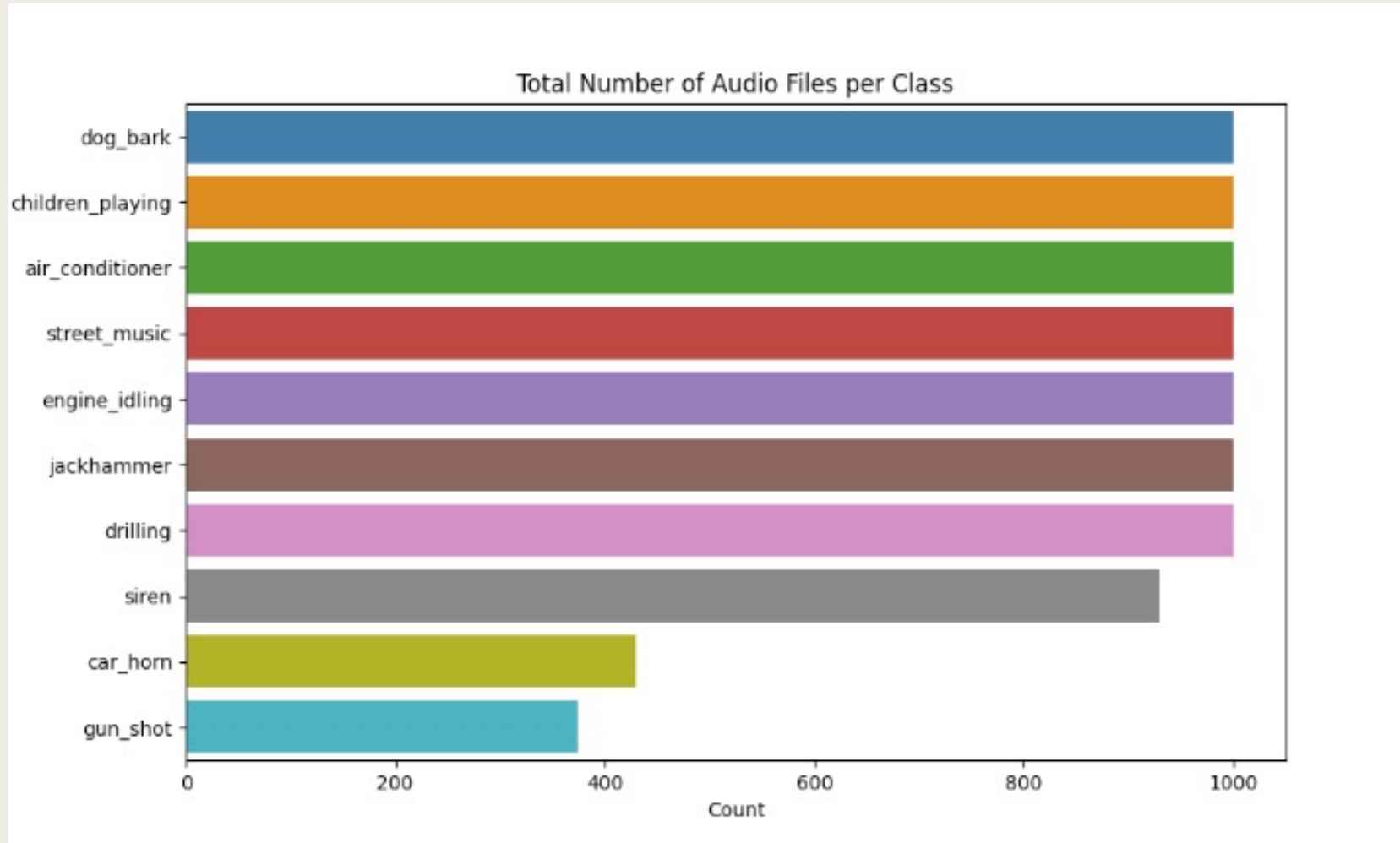
# Introduction

- UrbanSounds8K Dataset
  - 8732 labeled audio files (.wav format)
  - 10 classes
    - air\_conditioner, car\_horn, children\_playing, dog\_bark, drilling, engine\_idling, gun\_shot, jackhammer, siren, street\_music
  - Each audio file ~ 4 seconds (around 9.5 hours total)
  - Pre-sorted into 10 folds
  - Metadata included (.csv)
- Deep Speech topic
  - Network: CNN
  - Framework: PyTorch
    - Librosa, TorchAudio

# Data Analysis: Metadata

	slice_file_name	fsID	start	end	salience	fold	classID	class	num_channels	sampling_rate
0	177742-0-0-99.wav	177742	49.500000	53.500000	2	3	0	air_conditioner	2	48000
1	24074-1-0-10.wav	24074	18.060993	22.060993	1	1	1	car_horn	2	44100
2	60591-2-0-4.wav	60591	2.000000	6.000000	1	2	2	children_playing	2	44100
3	174026-3-1-5.wav	174026	11.719766	15.719766	2	4	3	dog_bark	2	48000
4	59594-4-0-3.wav	59594	1.500000	5.500000	2	2	4	drilling	2	44100
5	154758-5-0-7.wav	154758	3.500000	7.500000	1	4	5	engine_idling	2	48000
6	148841-6-2-0.wav	148841	9.132153	10.787741	1	5	6	gun_shot	2	44100
7	162134-7-7-0.wav	162134	129.628486	133.628486	1	10	7	jackhammer	2	96000
8	118279-8-0-13.wav	118279	6.500000	10.500000	2	1	8	siren	2	48000
9	89443-9-0-16.wav	89443	8.000000	12.000000	1	7	9	street_music	2	44100

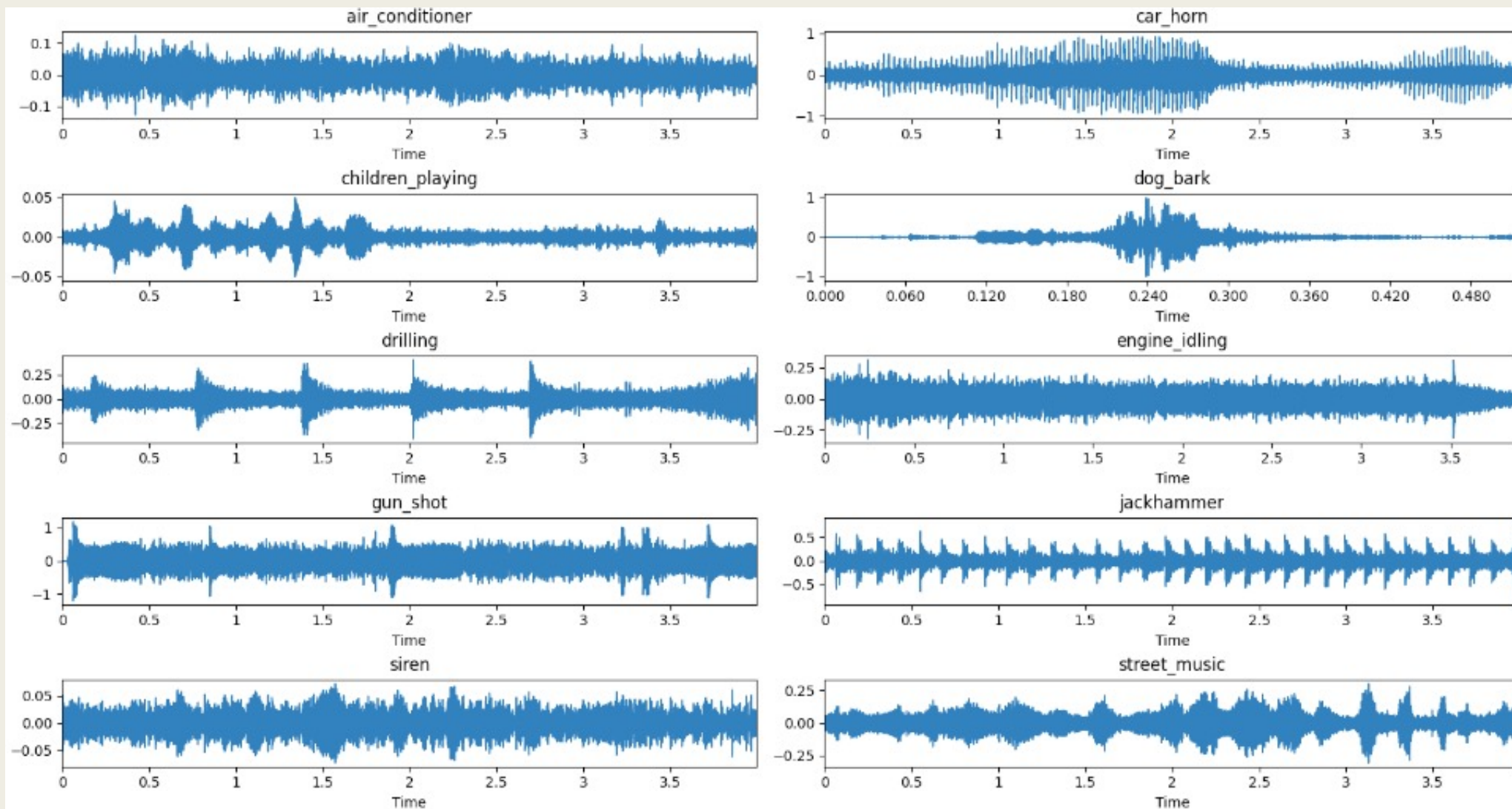
# Data Analysis: Audio Files per Class



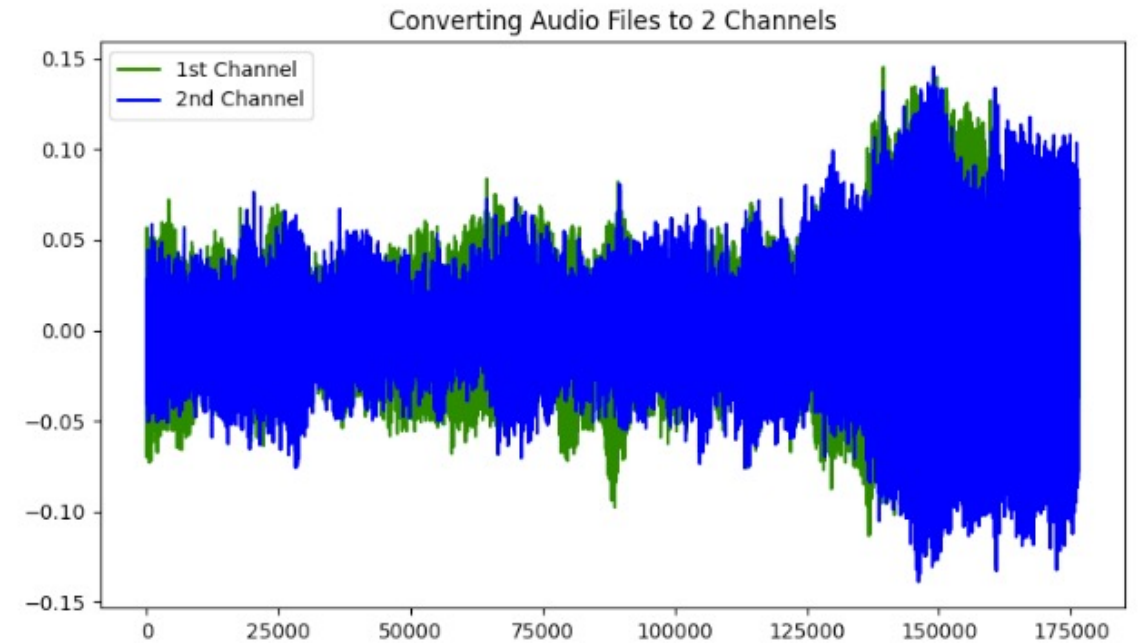
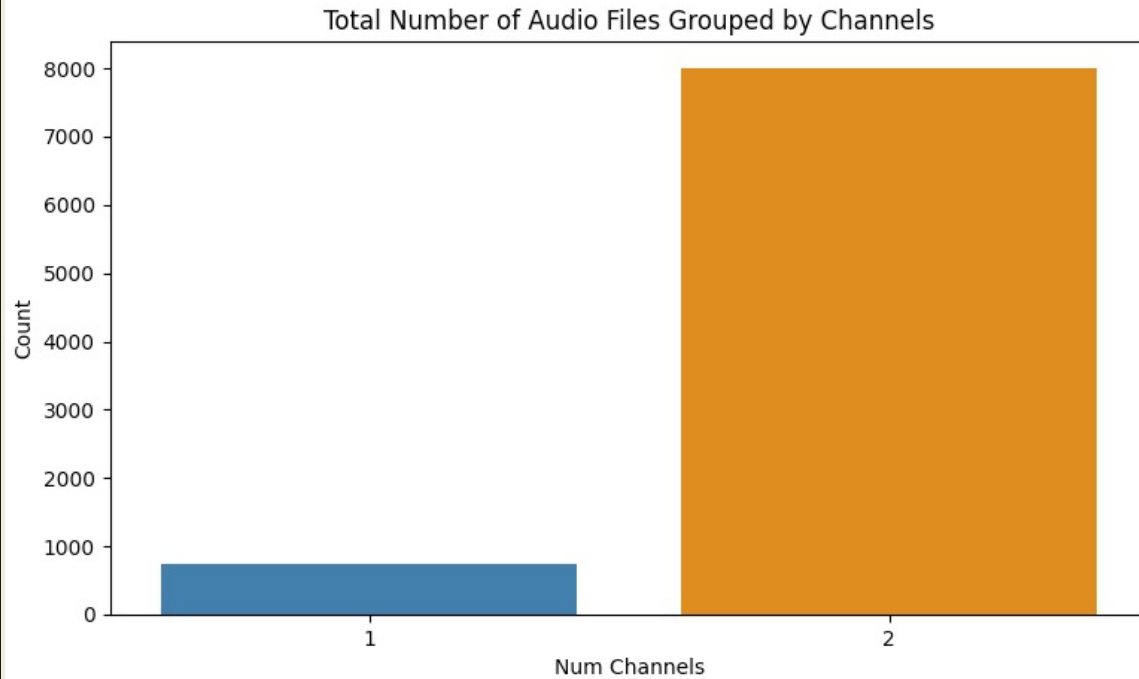
# Data Preprocessing

1. Load Audio File
2. Convert to Two Channels
3. Standardize Sampling Rate
4. Add Padding to Resize Audio to Same Length
5. Data Augmentation
  - Random Time Shift
  - Mel Spectrogram
  - Time & Frequency Masking

# 1. Load Audio Files



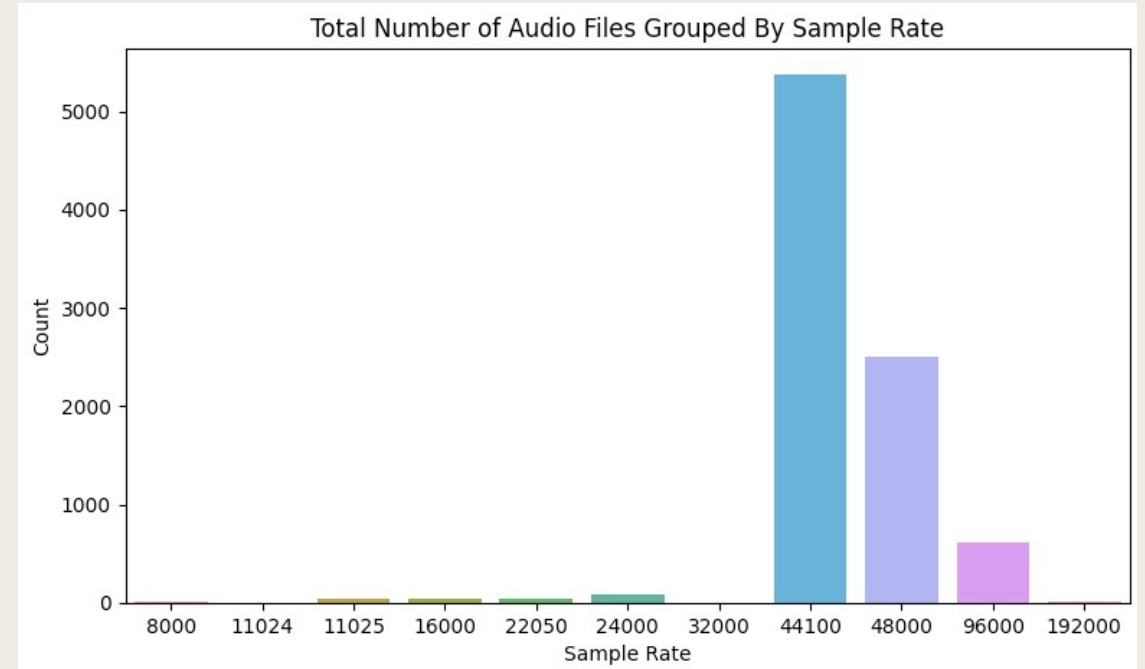
## 2. Convert to Two Channels



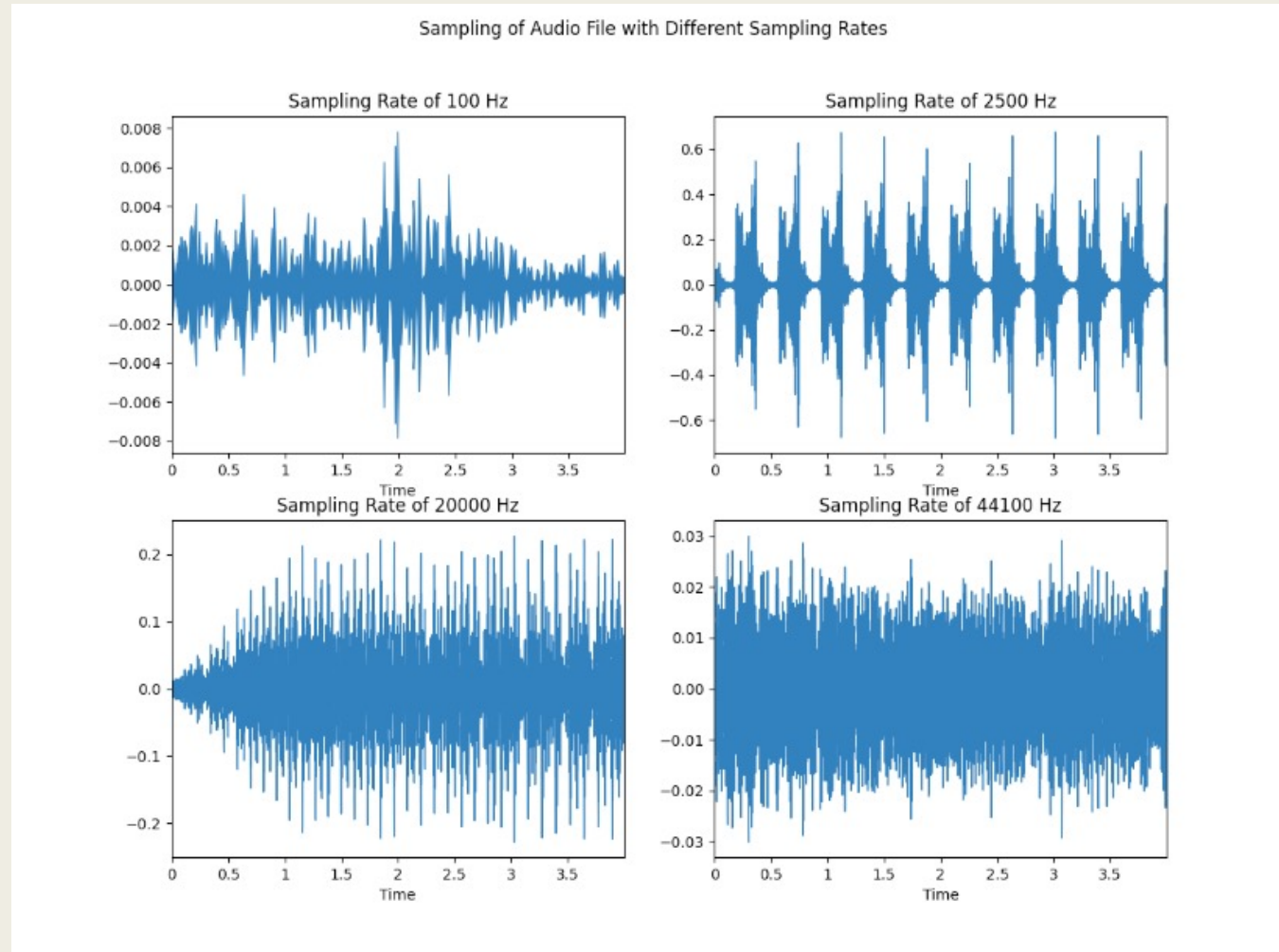


# 3. Standardize Sampling Rate

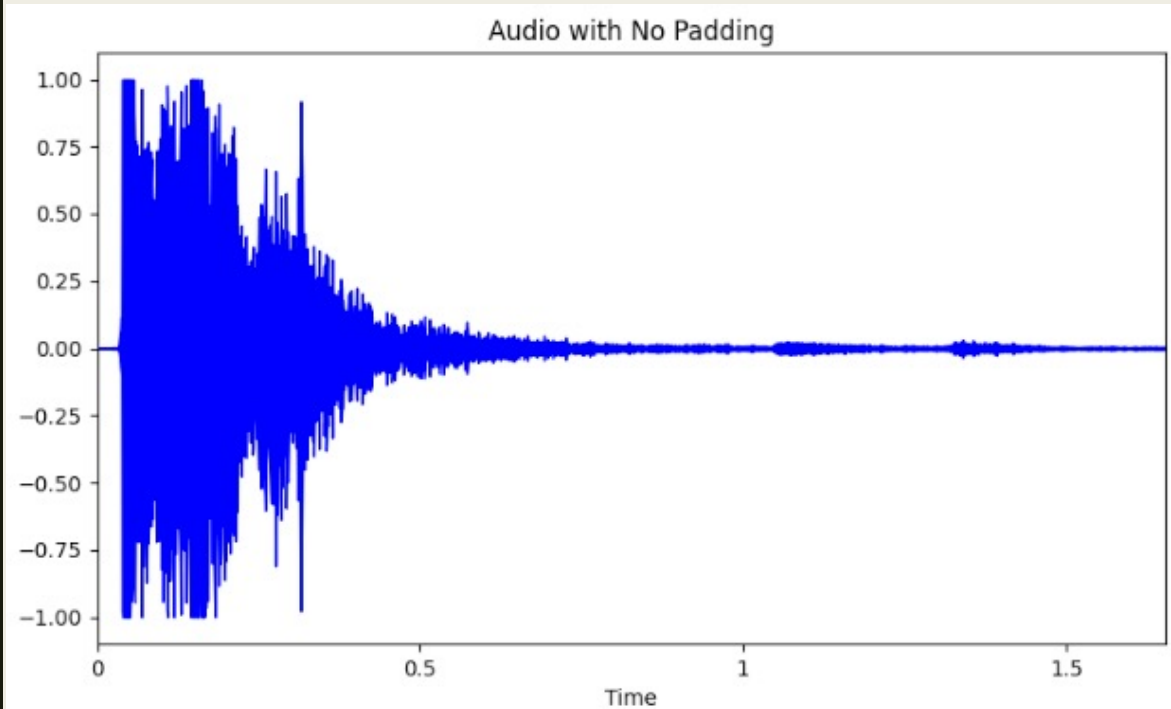
- Majority of Audio files are sampled at 44.1 kHz (44,100 Hz)
  - 1 second of audio will have an array of size 44,100
- Other sample rates will have dimensions of  $\text{num\_channels} \times (\text{sample\_rate} \times \text{time})$
- Standardize sampling rates of all audio files to 44.1 kHz



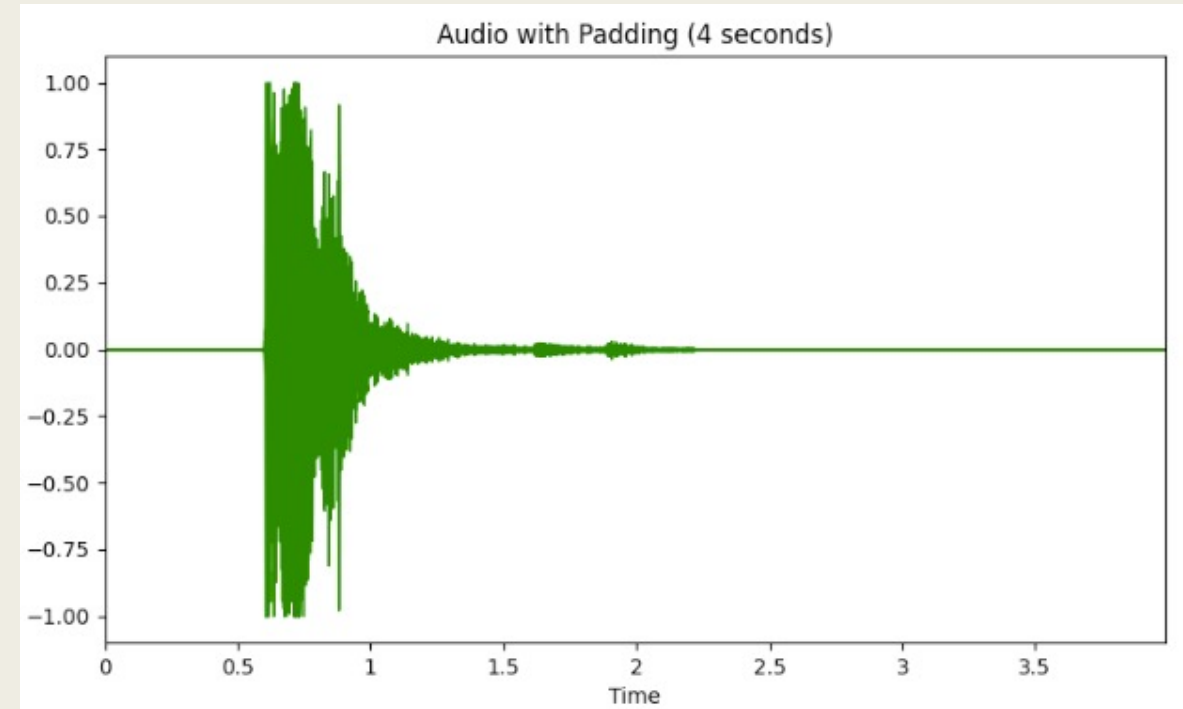
# 3. Standardize Sampling Rate



## 4. Add Padding to Resize Audio to Same Length



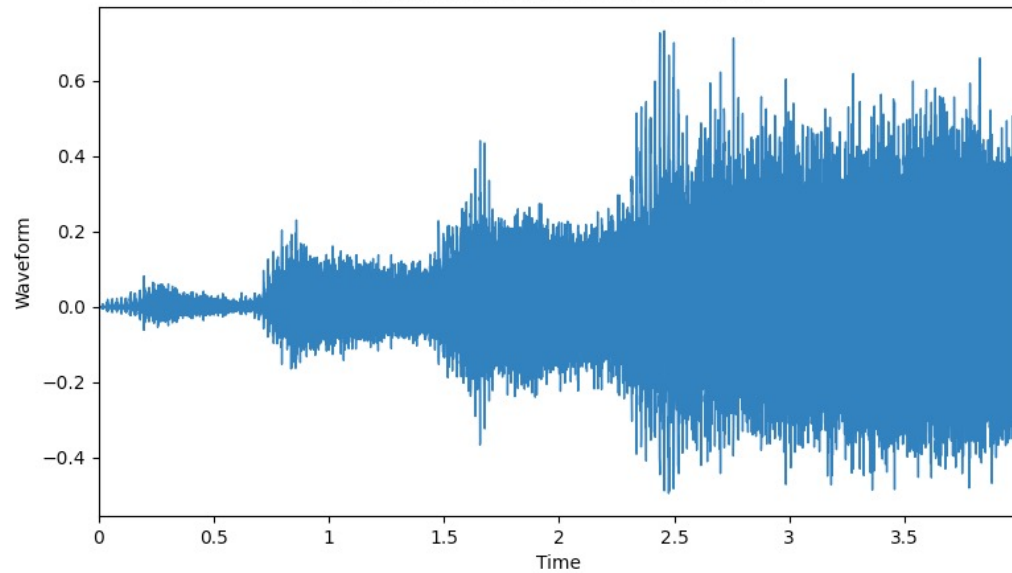
- Before Padding:
  - Time: ~ 1.75 seconds



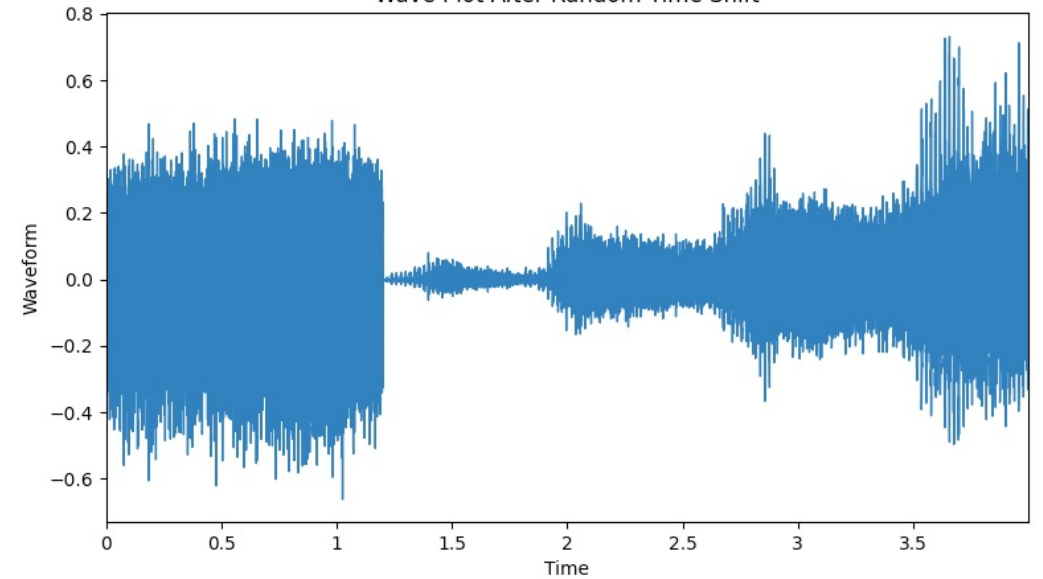
- After Padding:
  - Time: 4 seconds

# 5. Data Augmentation: Time Shift

Wave Plot Before Random Time Shift

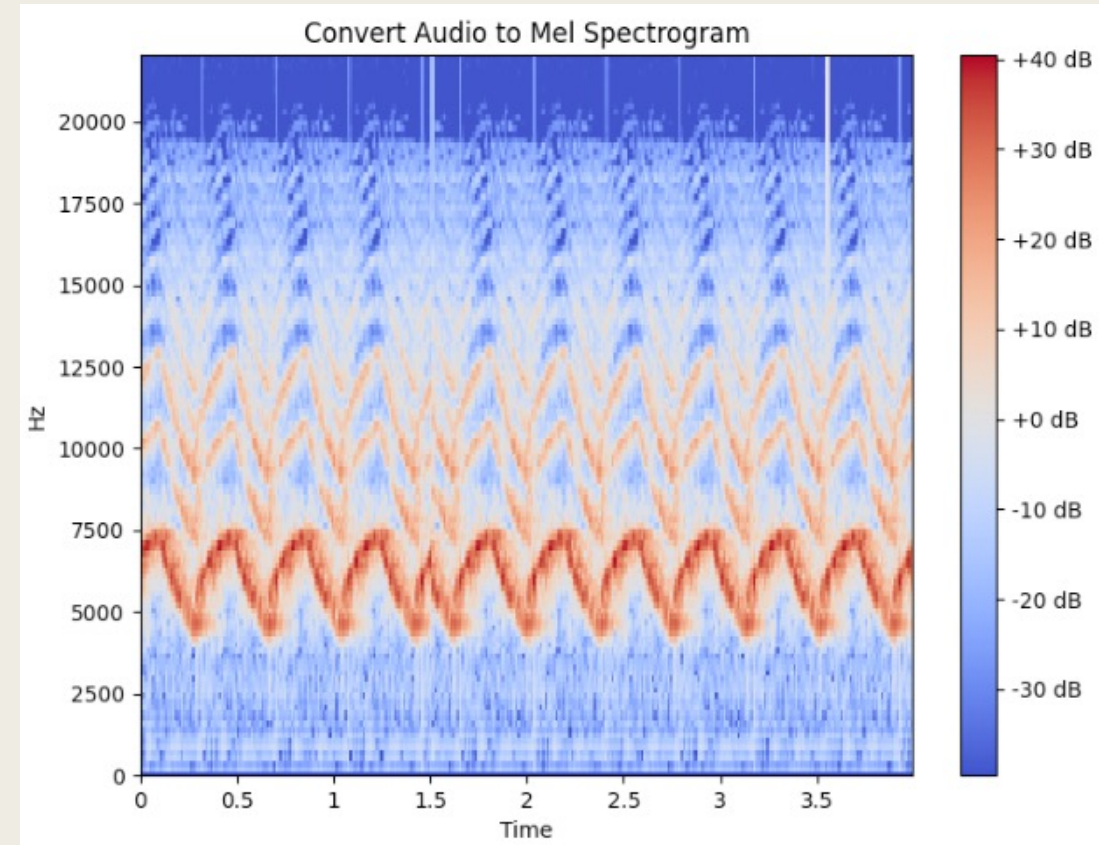


Wave Plot After Random Time Shift



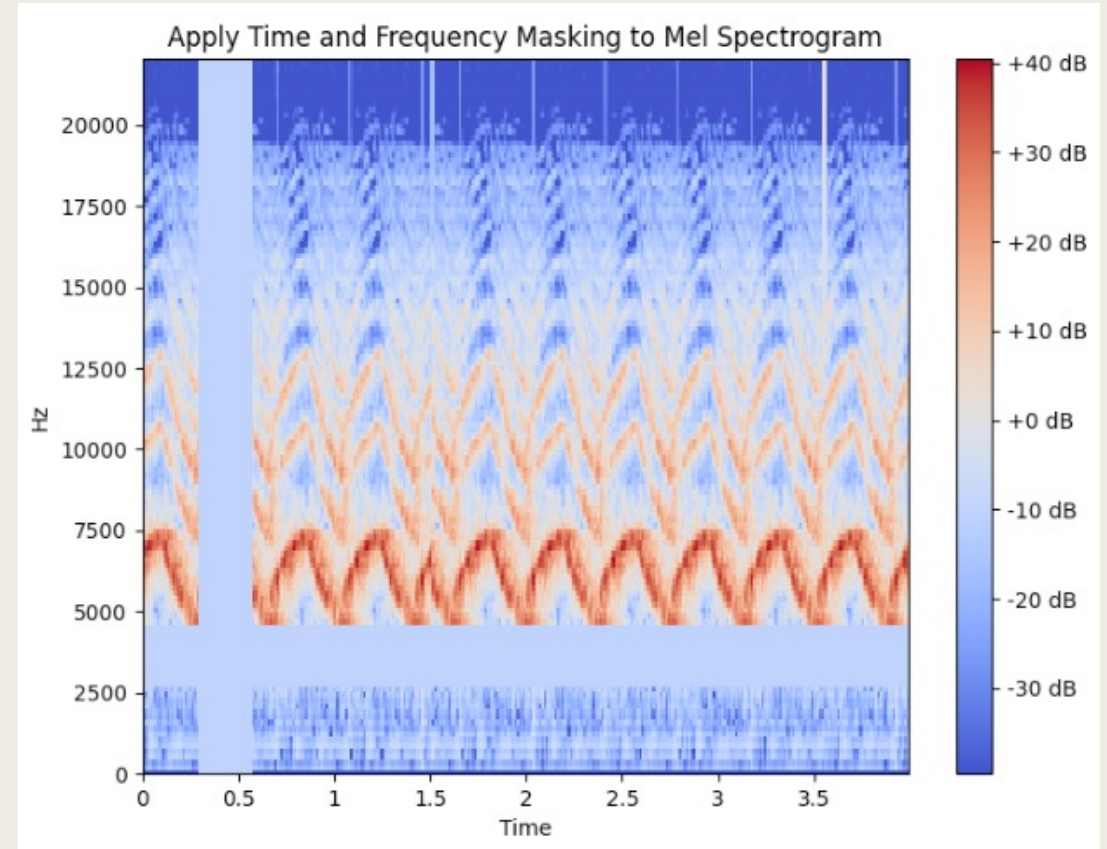
# 5. Data Augmentation: Mel Spectrogram

- Spectrograms
  - Chops up duration of a sound signal (waveform) into smaller segments
  - Plots Frequency (y-axis) vs. Time (x-axis)
  - Colors indicate amplitude of each frequency
- Mel Spectrograms
  - The Mel Scale instead of Frequency
  - Decibel Scale instead of Amplitude
    - Provides more useful information to a deep learning model



# 5. Data Augmentation: Time & Frequency Masking

- Frequency Masking
  - Randomly mask out a range of consecutive frequencies (horizontal bars)
- Time Masking
  - Similar to Frequency Masking, but instead masks out a range of time (vertical bars)





# CNN Architecture

## ■ 4-layer network

- Convolution-2D
  - Kernel Size = (5, 5) → (3, 3)
  - Stride = (2, 2)
  - Padding = (2, 2) → (1, 1)
- Batch Normalization
- Zero-Pad-2D
- Max-Pooling-2D
- Adaptive-Average-Pooling-2D
- Fully-Connected-Linear-Layer
- Activation Function: ReLU

```
AudioClassifier(  
  (conv1): Conv2d(2, 8, kernel_size=(5, 5), stride=(2, 2), padding=(2, 2))  
  (batch1): BatchNorm2d(8, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)  
  (pad1): ZeroPad2d(padding=(2, 2, 2, 2), value=0.0)  
  (pool1): MaxPool2d(kernel_size=5, stride=2, padding=0, dilation=1, ceil_mode=False)  
  (conv2): Conv2d(8, 32, kernel_size=(3, 3), stride=(2, 2), padding=(1, 1))  
  (batch2): BatchNorm2d(32, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)  
  (pad2): ZeroPad2d(padding=(2, 2, 2, 2), value=0.0)  
  (conv3): Conv2d(32, 64, kernel_size=(3, 3), stride=(2, 2), padding=(1, 1))  
  (batch3): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)  
  (conv4): Conv2d(64, 128, kernel_size=(3, 3), stride=(2, 2), padding=(1, 1))  
  (batch4): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)  
  (act): ReLU()  
  (pool2): AdaptiveAvgPool2d(output_size=1)  
  (linear1): Linear(in_features=128, out_features=10, bias=True)  
)
```

# CNN Architecture

- Input size: (batch\_size, num\_channels, mel\_freq, time\_steps)
  - (16, 2, 64, 344)
- Image width and height are reduced with kernels and strides
- After forward propagation through CNN layers, pooling layer, and linear layer, output is (batch\_size, num\_classes)
  - (16, 10)

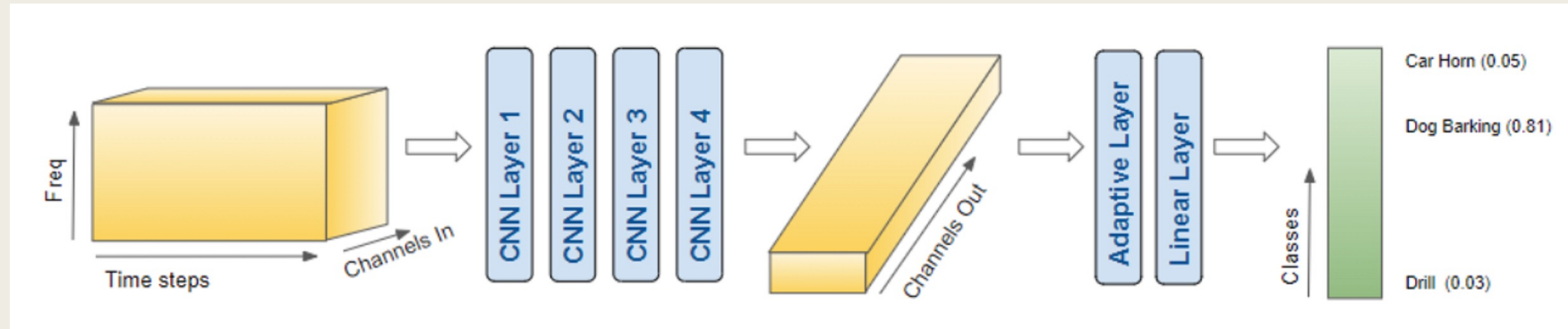


Image by: [Ketan Doshi](#)



# Training & Validation

## ■ Model Parameters:

- Epochs: 20
- Batch Size: 16
- Learning Rate: 0.001
- Optimizer: AdamW
- Scheduler: ReduceLROnPlateau
- Loss Function: CrossEntropyLoss

## ■ Training Results (70%):

- Accuracy: 91.95%
- Total Training Time: 39 min 33 secs
- Avg Time per Epoch: 1 min 58 secs

## ■ Validation Results (15%):

- Accuracy: 90.992%
- Total Validation Time: 7 min 58 secs
- Avg Time per Epoch: 24 secs

# Results – Testing the Model

- Testing Results (15%):
  - Accuracy: 91.145%
  - F1-Score: 0.9115
  - Total Testing Time: 21 seconds



# Future Work

- Apply Transformers to UrbanSounds9K
  - Wav2Vec2 (Hugging Face)
    - “A framework for self-supervised learning of speech representations.”
  - XLSR-Wav2Vec2 (Hugging Face)
    - “Unsupervised cross-lingual representation learning for speech recognition”

# Conclusion

- Ability to work with and preprocess Audio data (Librosa & TorchAudio)
  - Reading in audio as a wave-plot
  - Standardizing audio files (channels, sampling rate, padding time)
  - Convert wave-plot to Mel Spectrogram
  - Data Augmentation (random time shifts, frequency & time masking)
- Achieved high levels of accuracy with CNN
  - Training, Validation, Testing > 90% Accuracy
- Increased experience using PyTorch

# References

1. [UrbanSounds8K Official Website](#)
2. [UrbanSounds8K Data Download \(Kaggle\)](#)
3. [Audio Classification Analysis using Librosa \(GitHub\)](#)
4. [UrbanSound8K Audio Analysis](#)
5. [Audio Deep Learning Made Simple: Sound Classification](#)
6. [Audio Deep Learning: Why Mel Spectrograms Perform Better](#)
7. [UrbanSound8K Benchmarks \(PapersWithCode\)](#)