

Training Sound Classification Models in Create ML

Dan Klingler, Audio

Sound Classification

Sound Classification

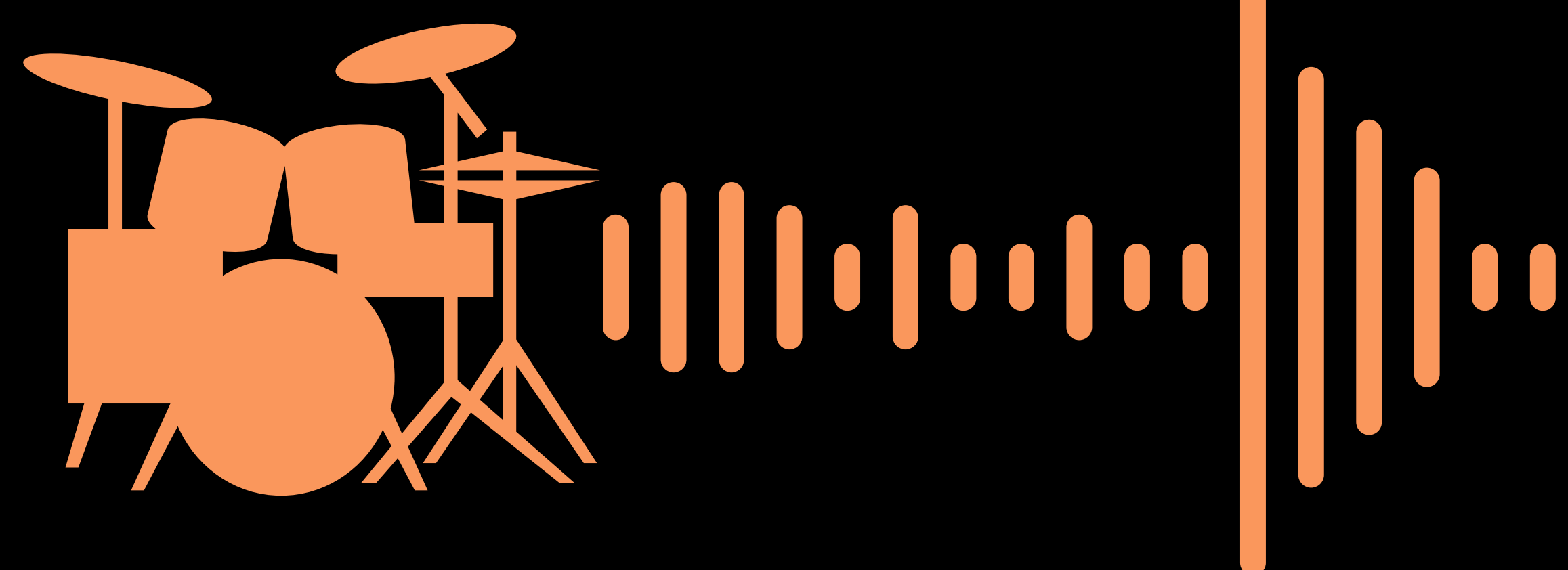


Sound Classification

Guitar

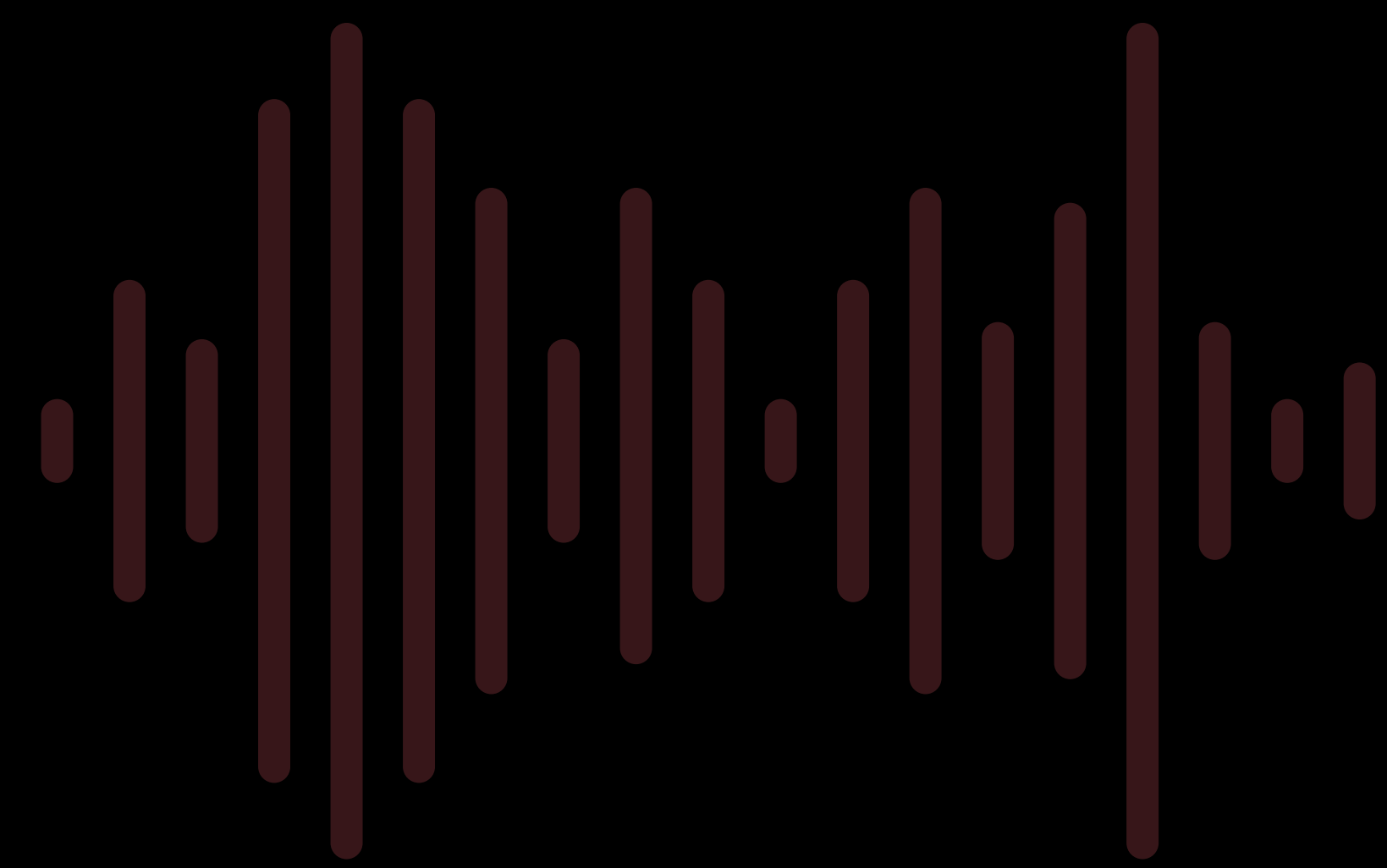


Drums



Sound Classification

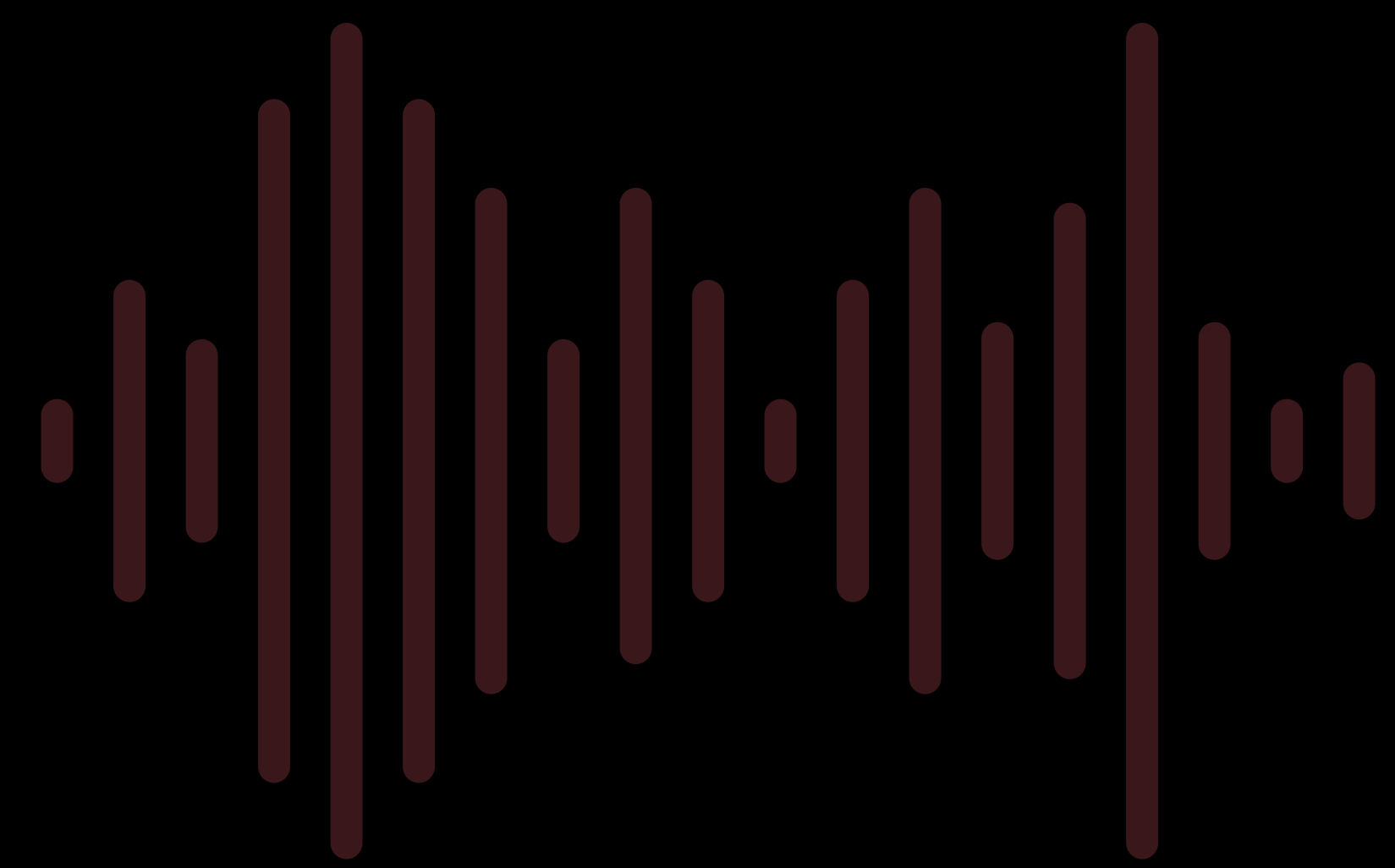
Nature



City



Sound Classification



Laugh

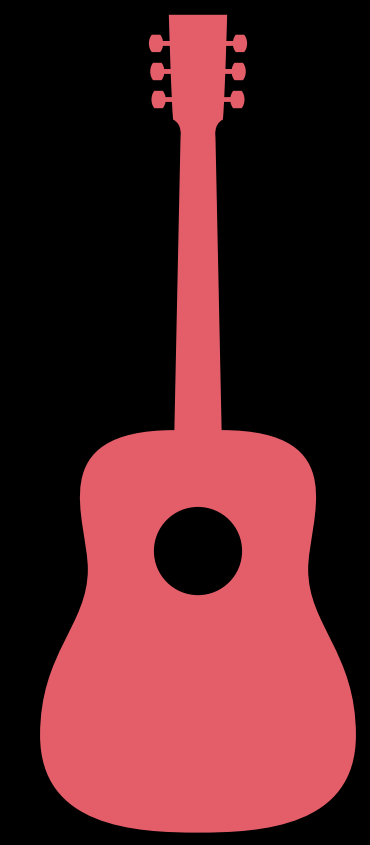


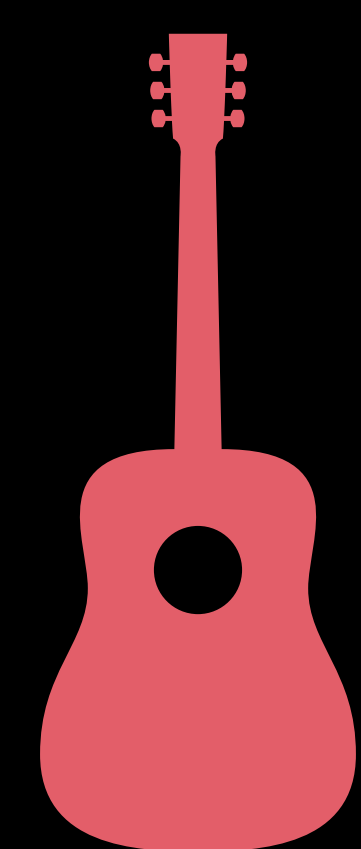
Cry

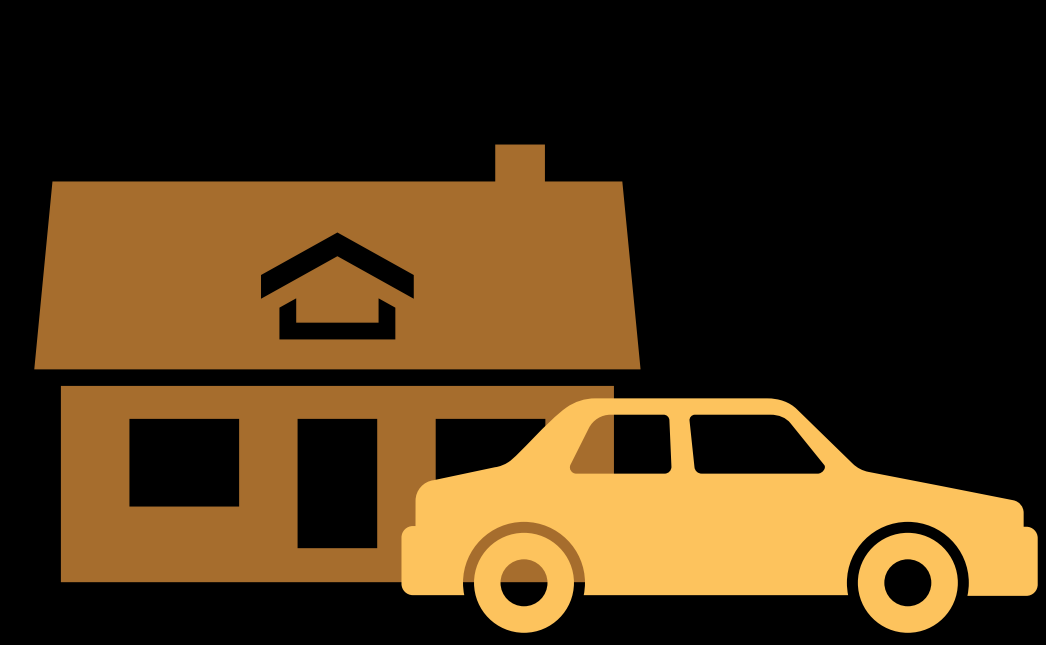
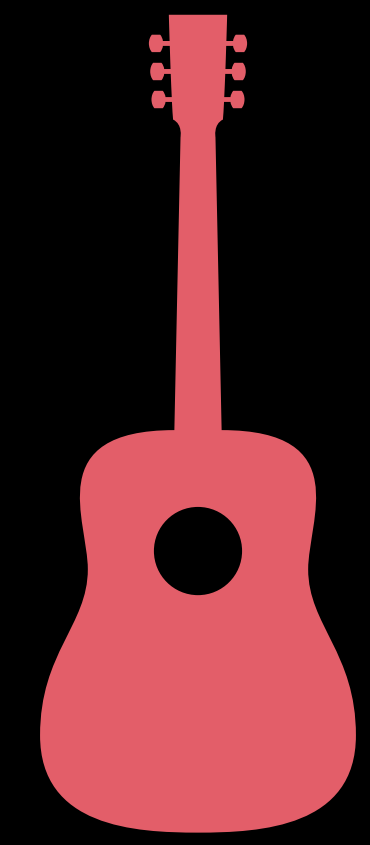




Create ML



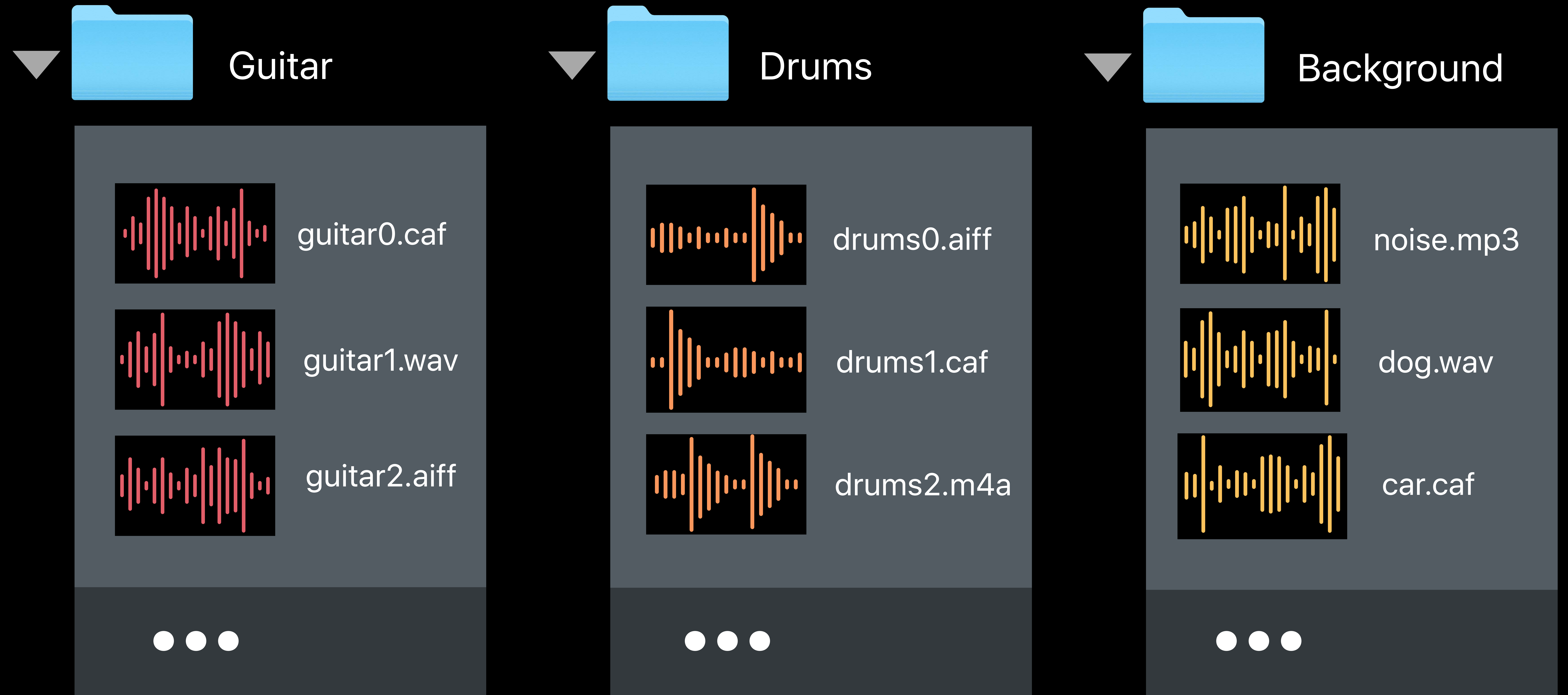




Demo

Training Data

Training Data



Considerations

Audio files

Considerations

Audio files

sounds.caf

Considerations

Audio files

sounds.caf



Considerations

Audio files

sounds.caf



Considerations

Audio files

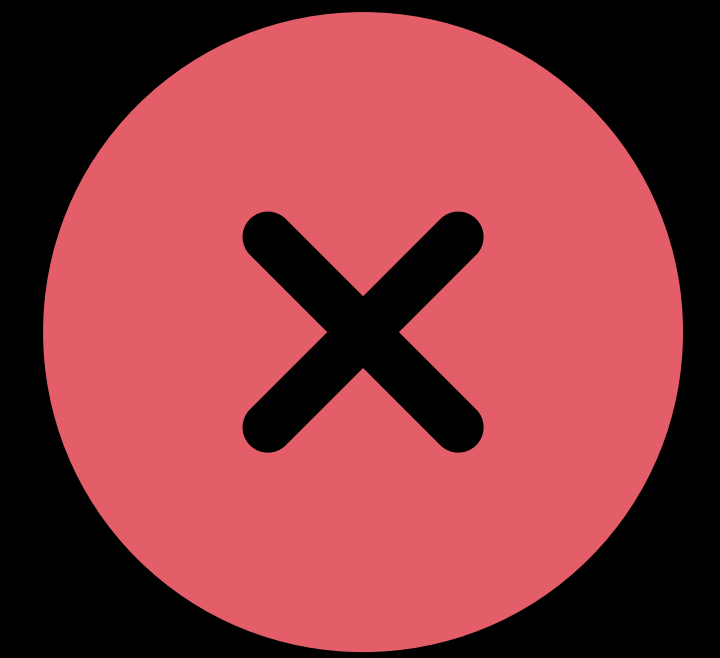
sounds.caf



Considerations

Audio files

sounds.caf



Considerations

Audio files

drums.caf



guitar.caf



background.caf



Considerations

Audio files

drums.caf



guitar.caf



background.caf



Considerations

Audio files

Considerations

Audio files

Ensure data matches real-world audio environments

Considerations

Audio files

Ensure data matches real-world audio environments

Consider device microphone processing

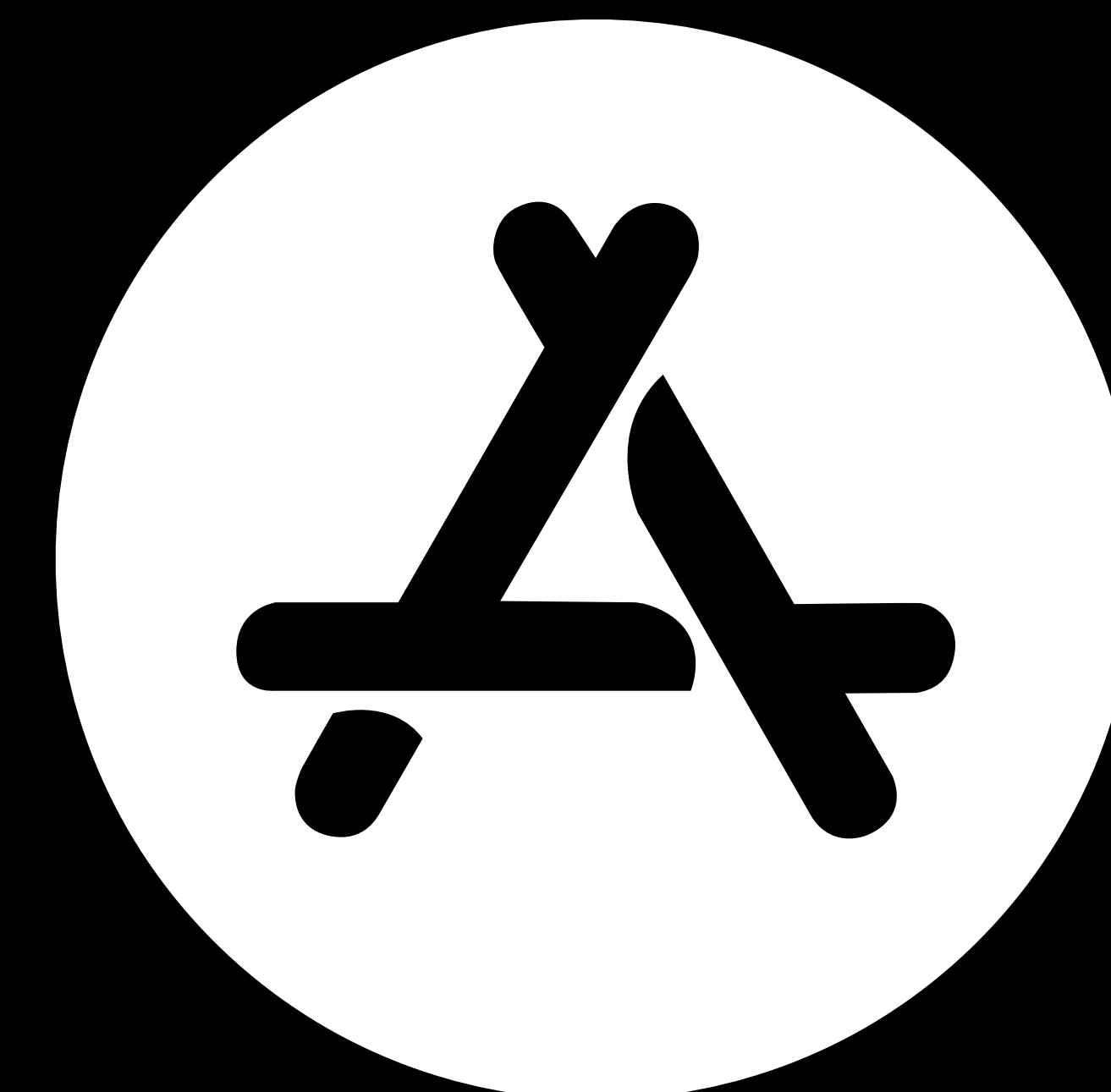
Considerations

Audio files

Ensure data matches real-world audio environments

Consider device microphone processing

Be aware of model architecture



Sound Analysis Framework

NEW

Sound Analysis Framework



NEW

New high-level framework for analyzing sound

Sound Analysis Framework



NEW

New high-level framework for analyzing sound

Uses Core ML sound classifier models

Sound Analysis Framework



NEW

New high-level framework for analyzing sound

Uses Core ML sound classifier models

Handles common audio operations internally

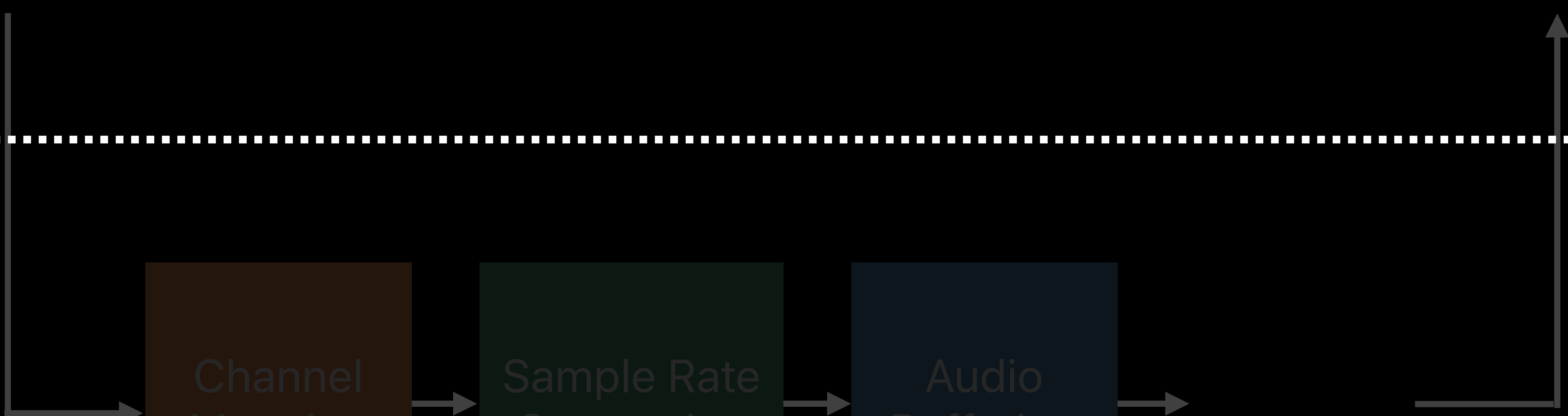
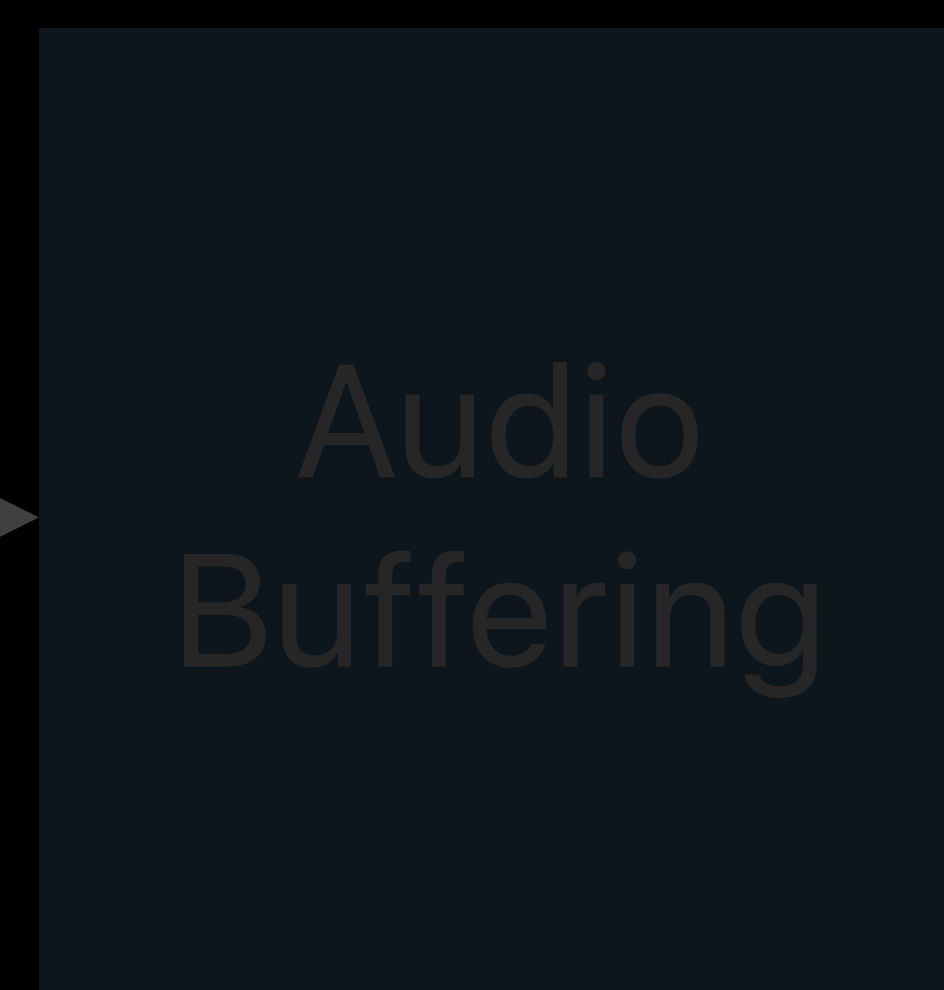
Sound Analysis Framework



Label	Confidence
Guitar	0.93
Background	0.05
Drums	0.02

Application

Sound Analysis



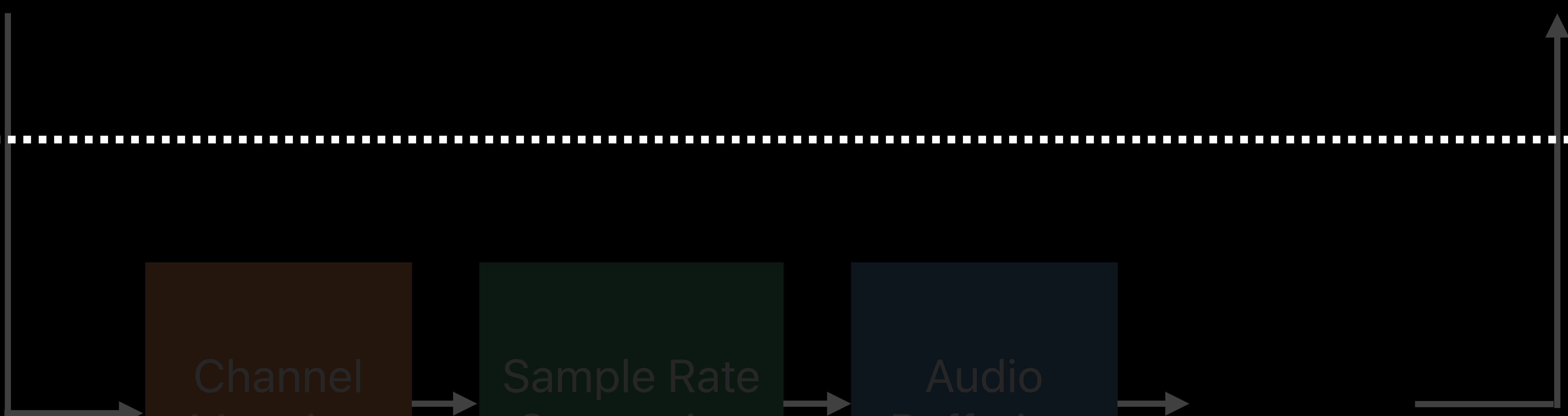
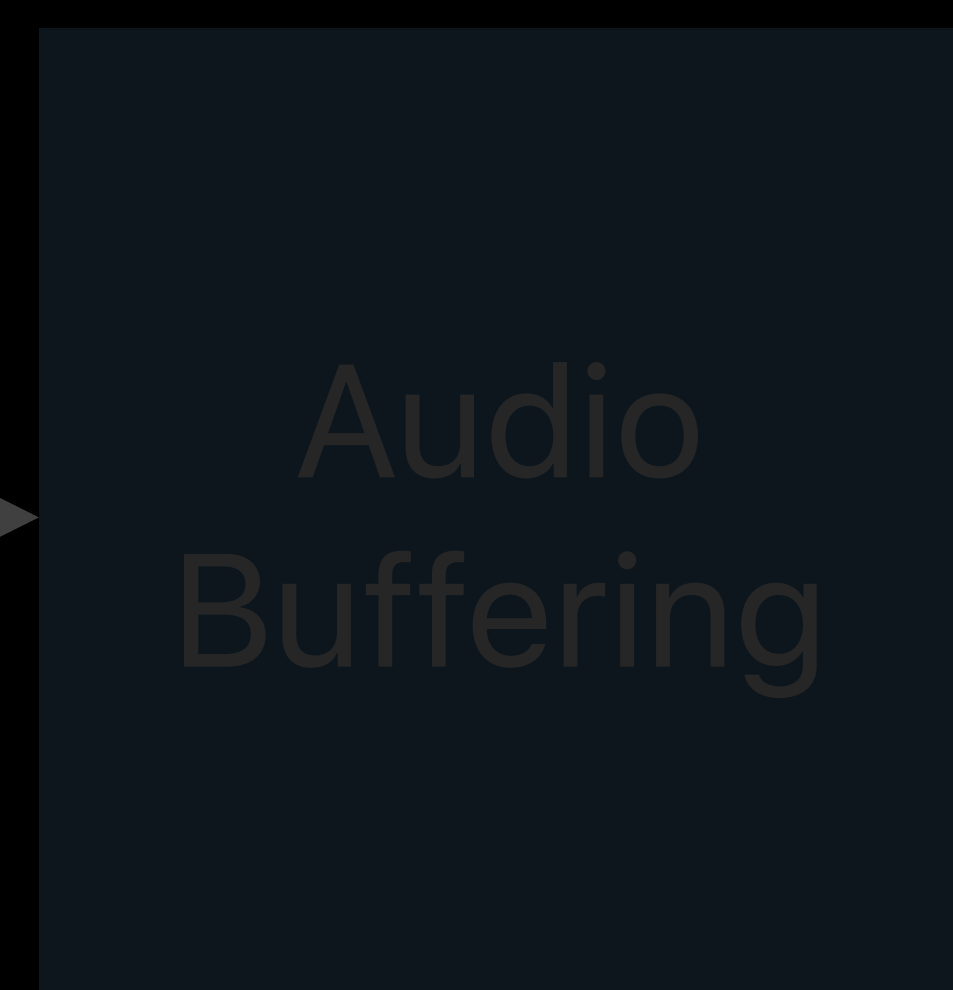
Sound Analysis Framework



Label	Confidence
Guitar	0.93
Background	0.05
Drums	0.02

Application

Sound Analysis



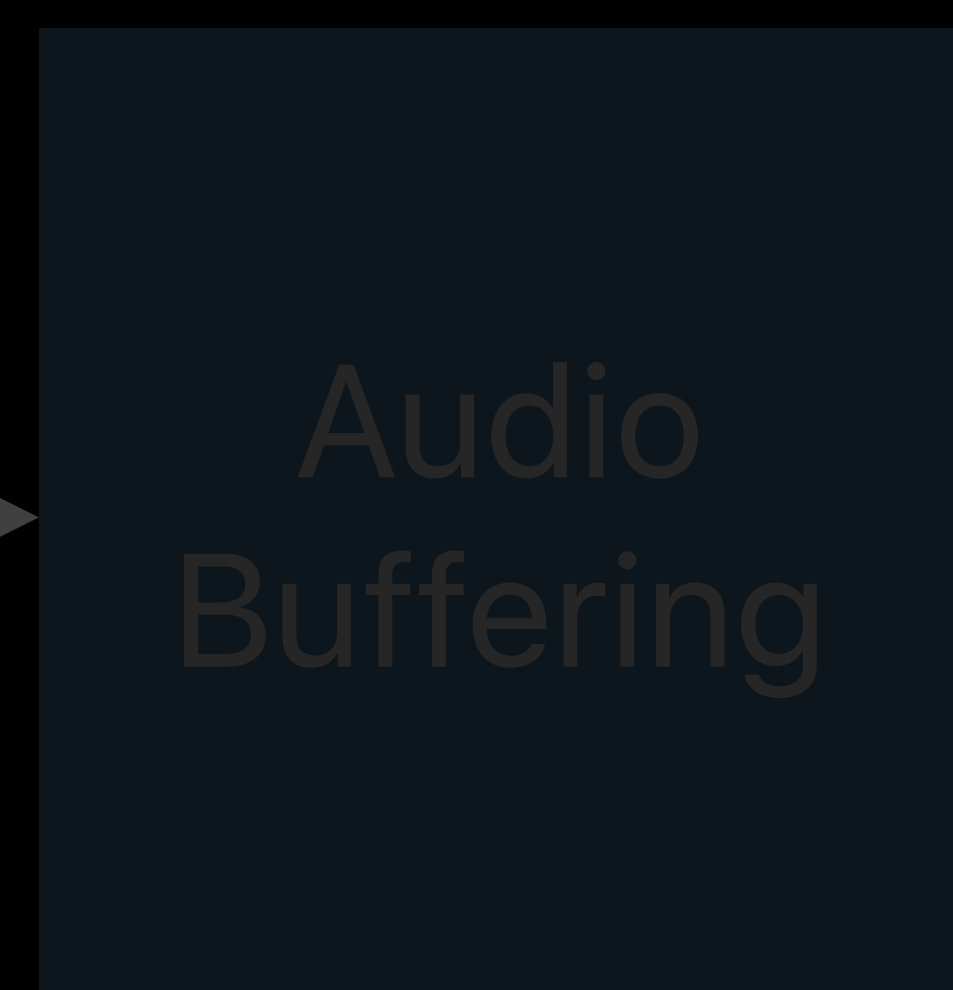
Sound Analysis Framework



Label		Confidence
Guitar	-->	0.93
Background	-->	0.05
Drums	-->	0.02

Application

Sound Analysis



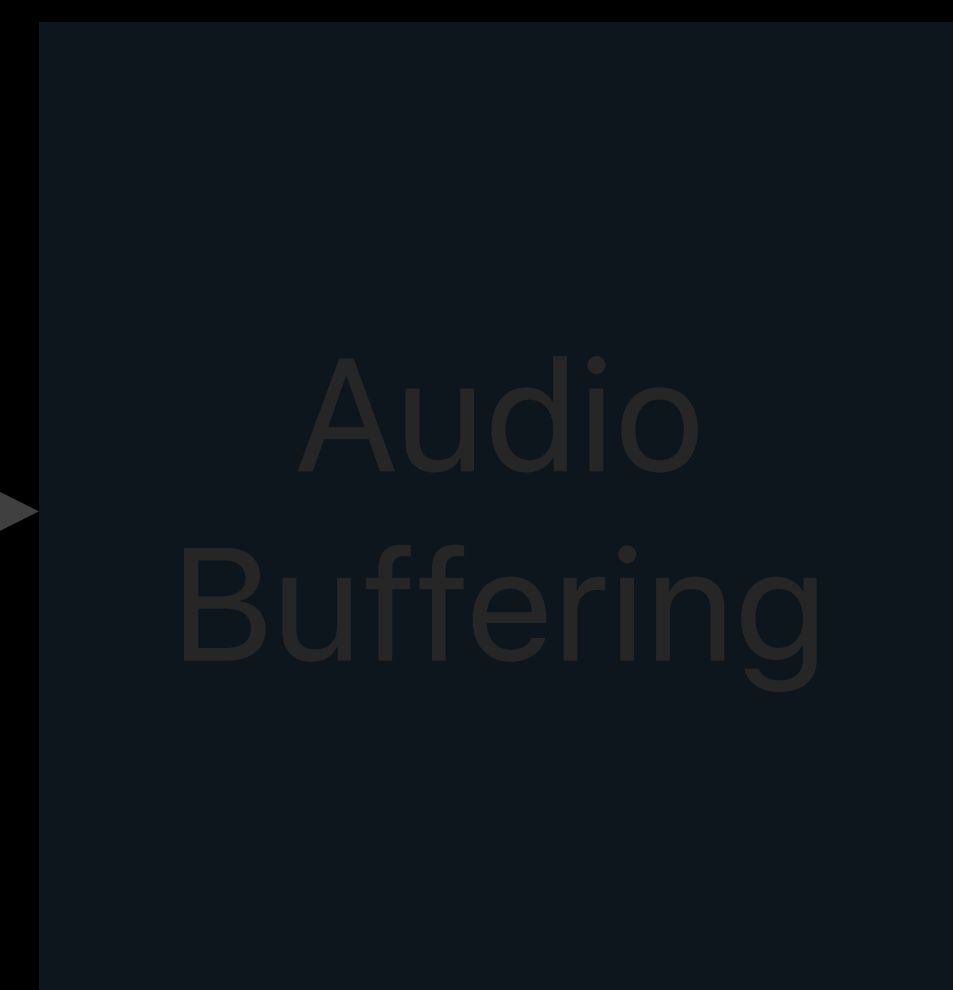
Sound Analysis Framework



Label		Confidence
Guitar	-->	0.93
Background	-->	0.05
Drums	-->	0.02

Application

Sound Analysis



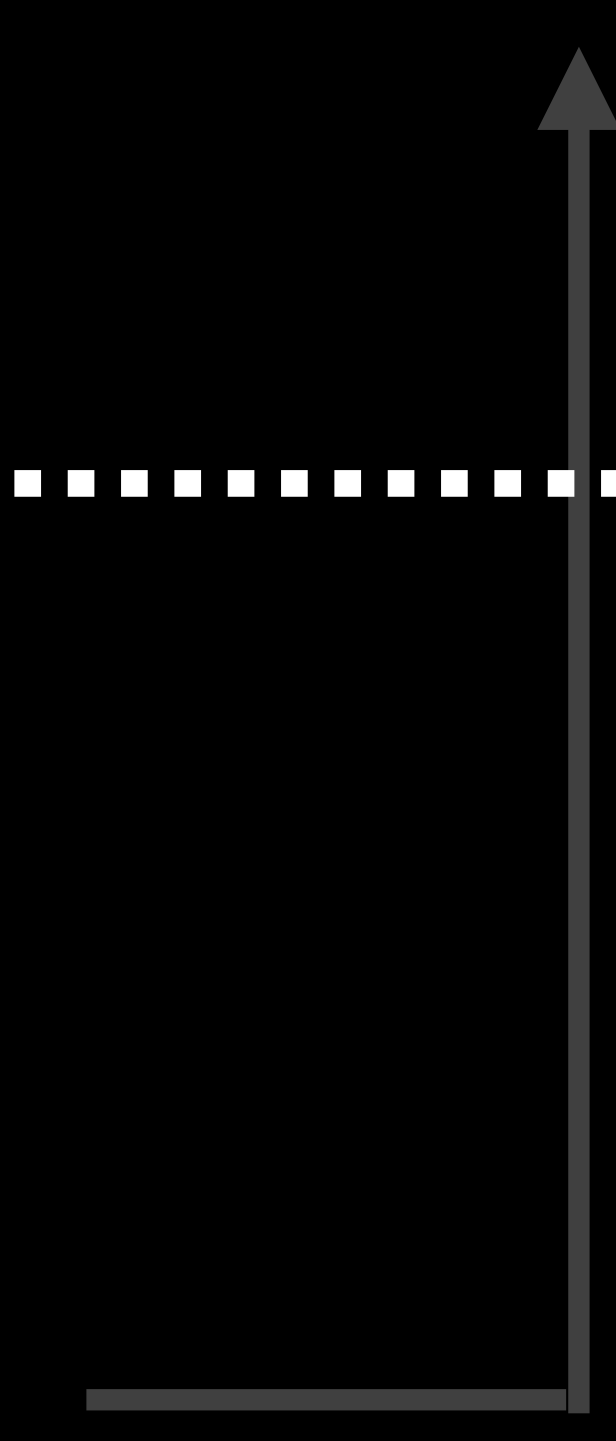
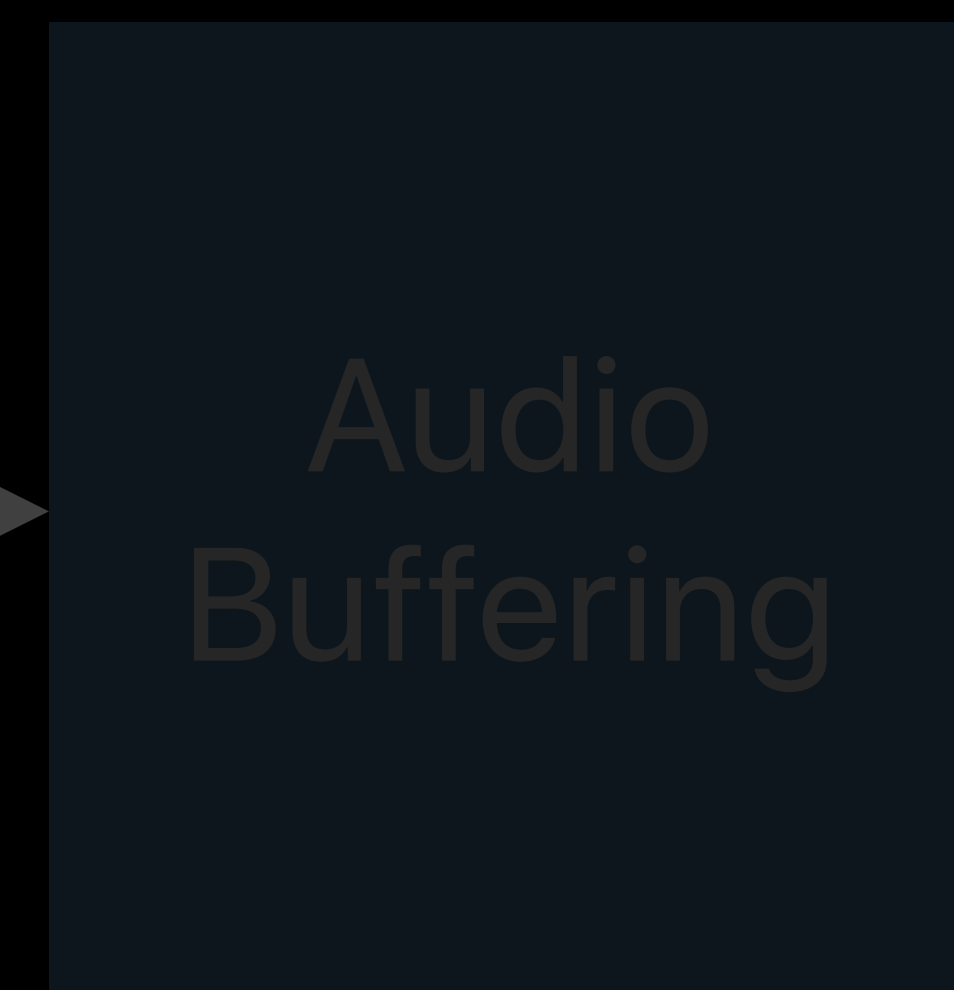
Sound Analysis Framework



Label		Confidence
Guitar	-->	0.93
Background	-->	0.05
Drums	-->	0.02

Application

Sound Analysis



Sound Analysis Framework



Label		Confidence
Guitar	-->	0.93
Background	-->	0.05
Drums	-->	0.02

Application

Sound Analysis



Sound Analysis Framework



Label		Confidence
Guitar	-->	0.93
Background	-->	0.05
Drums	-->	0.02

Application

Sound Analysis



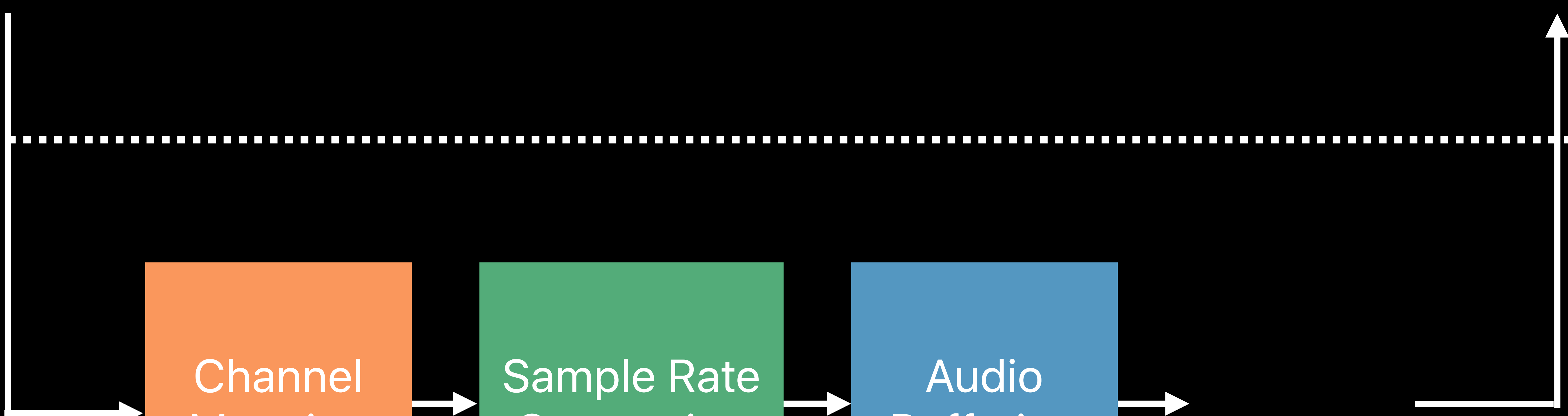
Sound Analysis Framework



Label		Confidence
Guitar	-->	0.93
Background	-->	0.05
Drums	-->	0.02

Application

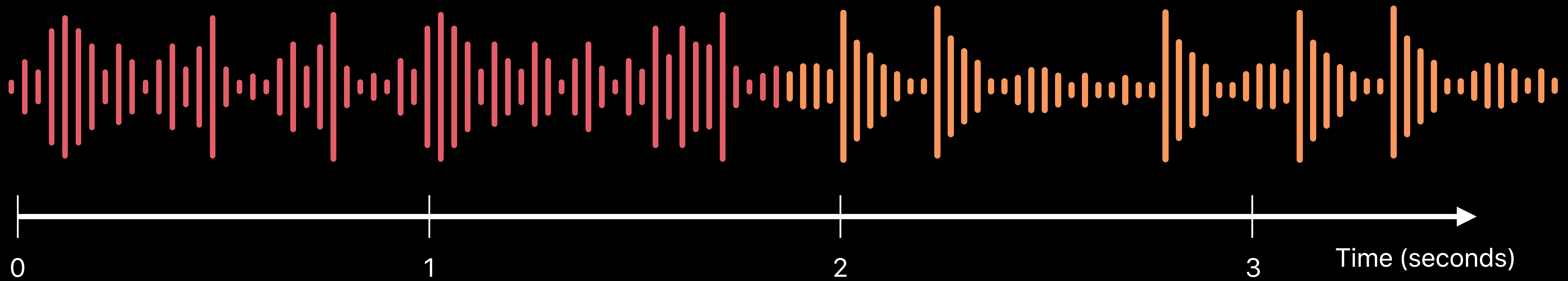
Sound Analysis



Sound Analysis Framework

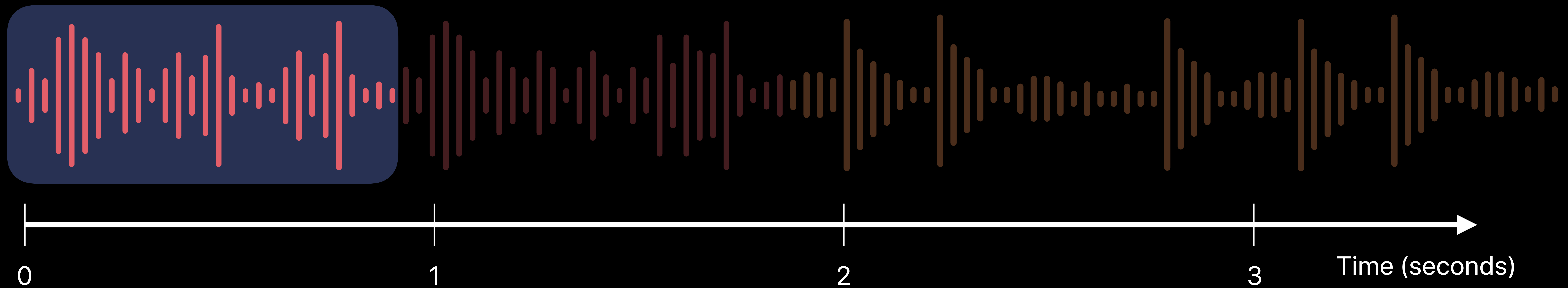


Results



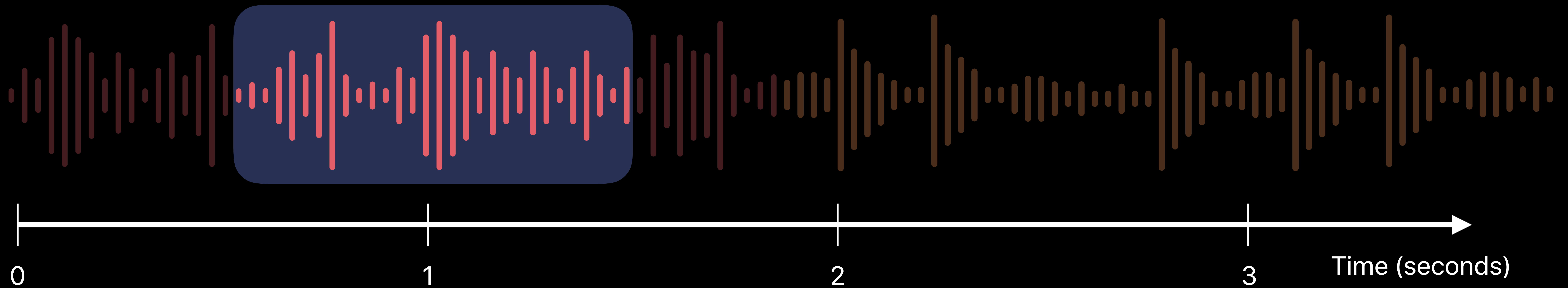
Results

Label		Confidence
Guitar	→	0.93
Background	→	0.05
Drums	→	0.02



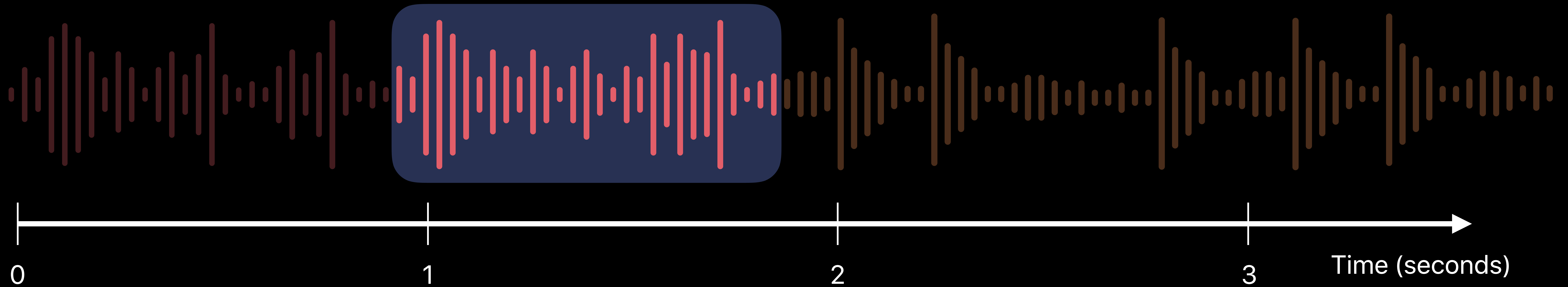
Results

Label		Confidence
Guitar	-->	0.95
Background	-->	0.03
Drums	-->	0.02



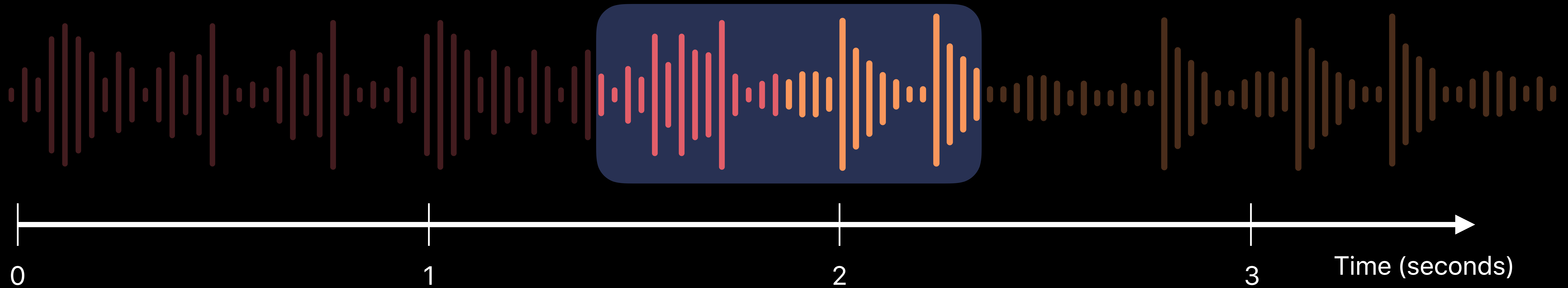
Results

Label		Confidence
Guitar	-->	0.91
Background	-->	0.05
Drums	-->	0.04



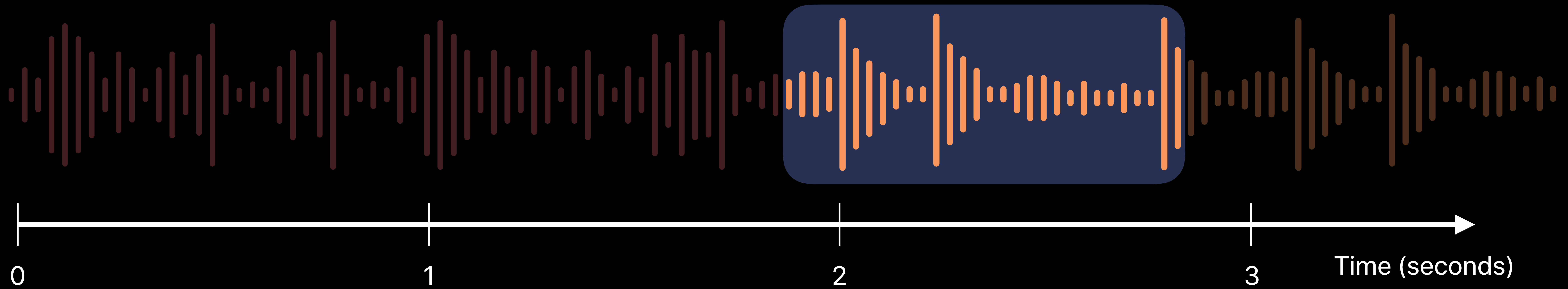
Results

Label		Confidence
Drums	-->	0.60
Guitar	-->	0.22
Background	-->	0.18



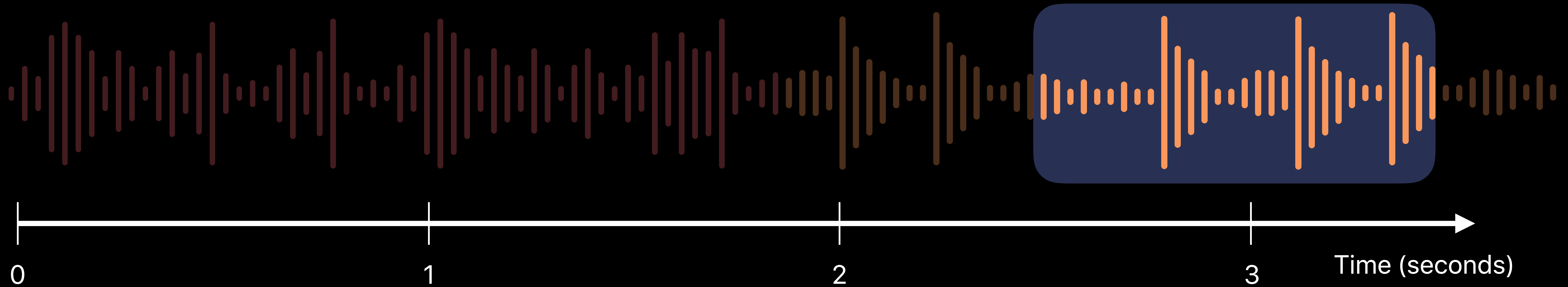
Results

Label	Confidence
Drums	→ 0.98
Background	→ 0.01
Guitar	→ 0.01



Results

Label	Confidence
Drums	0.96
Guitar	0.03
Background	0.01



Sound Analysis Framework

```
// Create a file analyzer
let fileAnalyzer = try SNAudioFileAnalyzer(url: URL(fileURLWithPath: "/Users/demo/sound.caf"))

// Create the request with your MLModel
let request = try SNClassifySoundRequest(mlModel: MySoundClassifier().model)

// Add the request to the analyzer
try fileAnalyzer.add(request, withObserver: self)

// Analyze the file
fileAnalyzer.analyze()
```

```
// Create a file analyzer
let fileAnalyzer = try SNAudioFileAnalyzer(url: URL(fileURLWithPath: "/Users/demo/sound.caf"))

// Create the request with your MLModel
let request = try SNClassifySoundRequest(mlModel: MySoundClassifier().model)

// Add the request to the analyzer
try fileAnalyzer.add(request, withObserver: self)

// Analyze the file
fileAnalyzer.analyze()
```

```
// Create a file analyzer
let fileAnalyzer = try SNAudioFileAnalyzer(url: URL(fileURLWithPath: "/Users/demo/sound.caf"))

// Create the request with your MLModel
let request = try SNClassifySoundRequest(mlModel: MySoundClassifier().model)

// Add the request to the analyzer
try fileAnalyzer.add(request, withObserver: self)

// Analyze the file
fileAnalyzer.analyze()
```

```
// Create a file analyzer
let fileAnalyzer = try SNAudioFileAnalyzer(url: URL(fileURLWithPath: "/Users/demo/sound.caf"))

// Create the request with your MLModel
let request = try SNClassifySoundRequest(mlModel: MySoundClassifier().model)

// Add the request to the analyzer
try fileAnalyzer.add(request, withObserver: self)

// Analyze the file
fileAnalyzer.analyze()
```

```
// Create a file analyzer
let fileAnalyzer = try SNAudioFileAnalyzer(url: URL(fileURLWithPath: "/Users/demo/sound.caf"))

// Create the request with your MLModel
let request = try SNClassifySoundRequest(mlModel: MySoundClassifier().model)

// Add the request to the analyzer
try fileAnalyzer.add(request, withObserver: self)

// Analyze the file
fileAnalyzer.analyze()
```

```
extension MyResultsObserver : SNResultsObserving {

    func request(_ request: SNRequest, didProduce result: SNResult) {
        guard let classificationResult = result as SNClassificationResult else { return }
        let topClassification = classificationResult.classifications.first
        let timeRange = classificationResult.timeRange
        // Handle result
    }

    func request(_ request: SNRequest, didFailWithError error: Error) {
        // Handle error
    }

    func requestDidComplete(_ request: SNRequest) {
        // Handle successful end of analysis
    }

}
```

```
extension MyResultsObserver : SNResultsObserving {
```

```
    func request(_ request: SNRequest, didProduce result: SNResult) {  
        guard let classificationResult = result as SNClassificationResult else { return }  
        let topClassification = classificationResult.classifications.first  
        let timeRange = classificationResult.timeRange  
        // Handle result  
    }
```

```
    func request(_ request: SNRequest, didFailWithError error: Error) {  
        // Handle error  
    }
```

```
    func requestDidComplete(_ request: SNRequest) {  
        // Handle successful end of analysis  
    }
```

```
}
```



```
extension MyResultsObserver : SNResultsObserving {
```

```
    func request(_ request: SNRequest, didProduce result: SNResult) {  
        guard let classificationResult = result as SNClassificationResult else { return }  
        let topClassification = classificationResult.classifications.first  
        let timeRange = classificationResult.timeRange  
        // Handle result  
    }
```

```
    func request(_ request: SNRequest, didFailWithError error: Error) {  
        // Handle error  
    }
```

```
    func requestDidComplete(_ request: SNRequest) {  
        // Handle successful end of analysis  
    }
```

```
}
```

```
extension MyResultsObserver : SNResultsObserving {

    func request(_ request: SNRequest, didProduce result: SNResult) {
        guard let classificationResult = result as SNClassificationResult else { return }
        let topClassification = classificationResult.classifications.first
        let timeRange = classificationResult.timeRange
        // Handle result
    }

    func request(_ request: SNRequest, didFailWithError error: Error) {
        // Handle error
    }

    func requestDidComplete(_ request: SNRequest) {
        // Handle successful end of analysis
    }

}
```

```
extension MyResultsObserver : SNResultsObserving {  
  
    func request(_ request: SNRequest, didProduce result: SNResult) {  
        guard let classificationResult = result as SNClassificationResult else { return }  
        let topClassification = classificationResult.classifications.first  
        let timeRange = classificationResult.timeRange  
        // Handle result  
    }  
  
    func request(_ request: SNRequest, didFailWithError error: Error) {  
        // Handle error  
    }  
  
    func requestDidComplete(_ request: SNRequest) {  
        // Handle successful end of analysis  
    }  
  
}
```

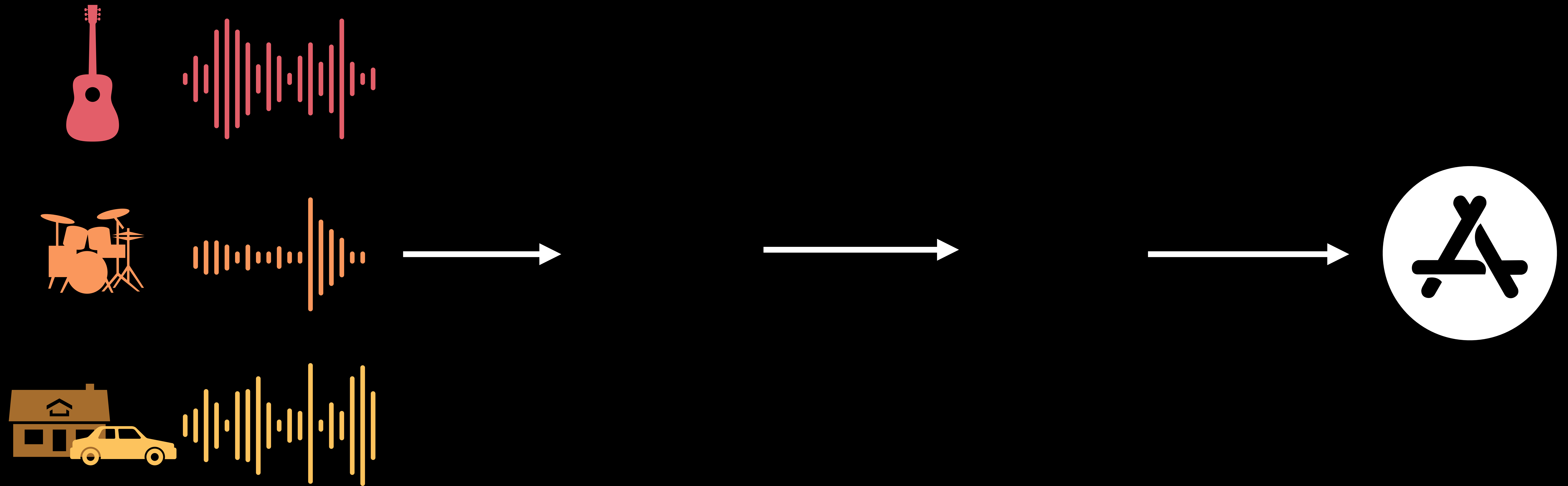
Recap

Recap

Train a sound classifier in Create ML using your own audio data

Recap

Train a sound classifier in Create ML using your own audio data



Summary

Run your model on-device using Sound Analysis



More Information

developer.apple.com/wwdc19/425

Create ML for Activity, Text, and Recommendations

Thursday, 2:00

Machine Learning Labs

Daily

