

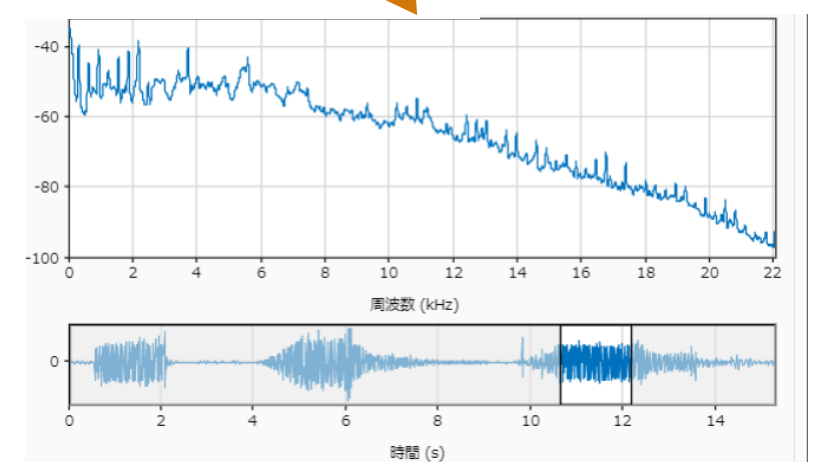
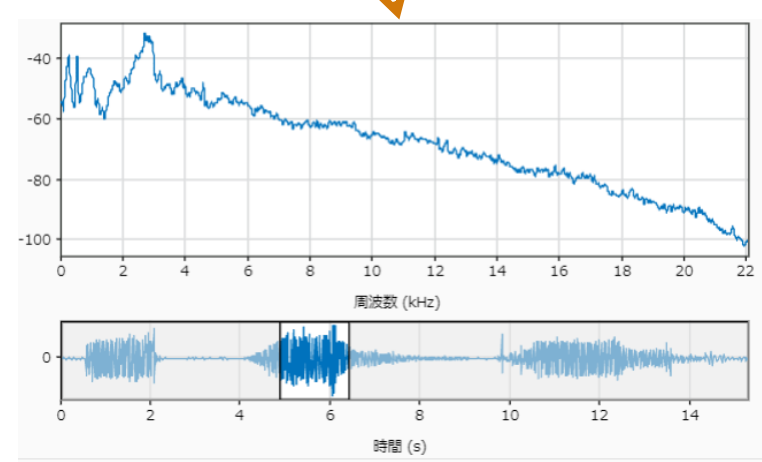
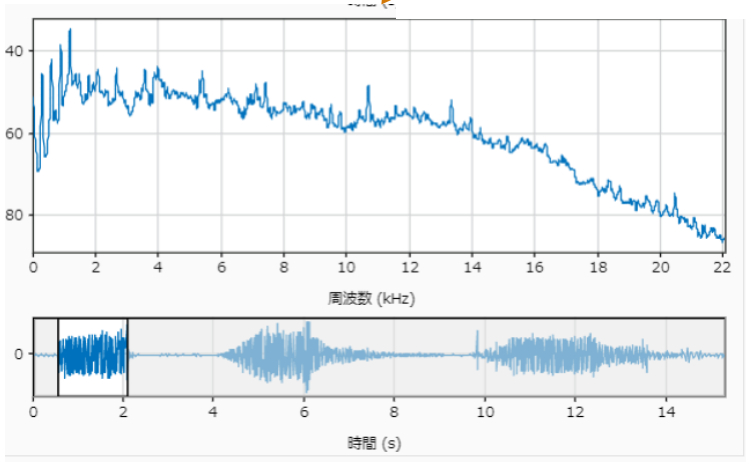
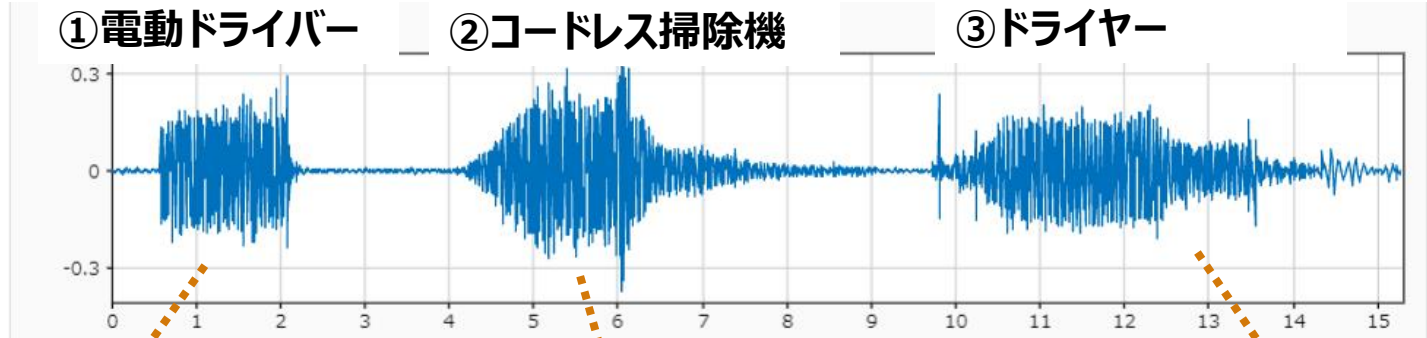
アプリで楽するMATLAB信号処理 ～AIへの応用を想定した活用例～

MathWorks Japan
アプリケーションエンジニアリング部
竹本佳充

何の音でしょうか？



①電動ドライバー ②コードレス掃除機 ③ドライヤー



AIシステム開発のワークフロー

1. データセットの作成

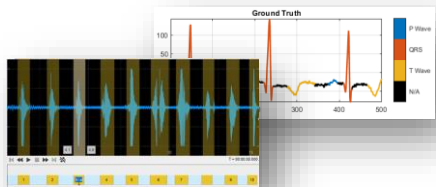
■ データソース



■ シミュレーション ■ オグメンテーション

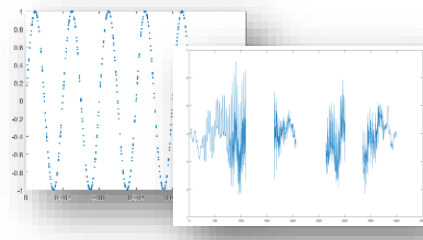


■ データラベリング

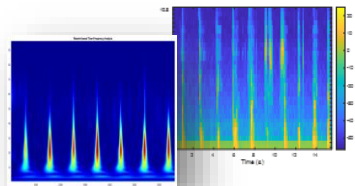


2. 前処理と変換

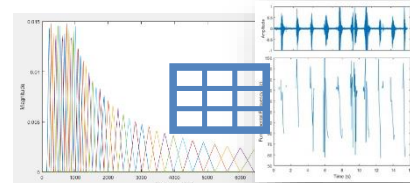
■ 前処理



■ 変換



■ 特徴抽出

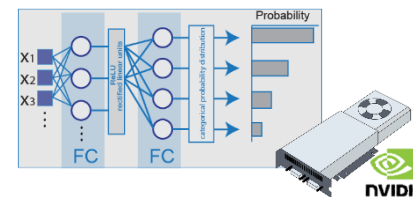


3. 予測モデルの開発

■ リファレンスモデルの流用 ■ オリジナルモデルの作成



■ GPUによる高速な学習

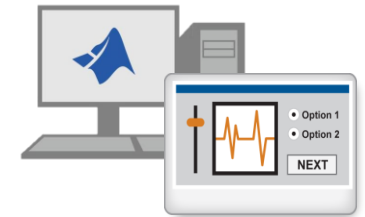


■ ハイパーパラメータの 解析とチューニング



4. 実システムへの展開

■ デスクトップアプリ



■ エンタープライズシステム

Java
MATLAB
C/C++
Python

■ エッジ（組み込み）デバイス



本セミナーのゴール：8種類の信号を自動判別する ～エアコンプレッサーの異音判定～

■ 異常項目

- ベアリング異常
- フライホイール異常
- インレットバルブ漏れ異常
- アウトレットバルブ漏れ異常
- ノンリターンバルブ異常
- ピストンリング異常
- ライダーベルト異常

■ 観測条件

- Air Pressure Range: 0-500 lb/m2, 0-35 Kg/cm2
- Induction Motor: 5HP, 415V, 5Am, 50 Hz, 1440rpm
- Pressure Switch: Type PR-15, Range 100-213 PSI



今回使用する主なアプリ



機械学習および
深層学習
関連アプリ



ディープ ネットワーク
デザイナー

テスト計測
関連アプリ



Raspberry Pi
リソース モニター

信号処理と通信



信号処理
関連アプリ

Agenda

- はじめに
- デモ
 1. データセットの準備 : **音の録音・再生**を楽しみたい
 2. 前処理・特徴抽出 : **波形の表示・解析**を楽しみたい
 3. 予測モデルの開発 : **深層ネットワークの編集**を楽しみたい
 4. 実システムへの展開 : **エッジ開発**を楽しみたい
- まとめ

Agenda

- はじめに
- デモ
 1. データセットの準備：音の録音・再生を楽しみたい
 2. 前処理・特徴抽出：波形の表示・解析を楽しみたい
 3. 予測モデルの開発：深層ネットワークの編集を楽しみたい
 4. 実システムへの展開：エッジ開発を楽しみたい
- まとめ



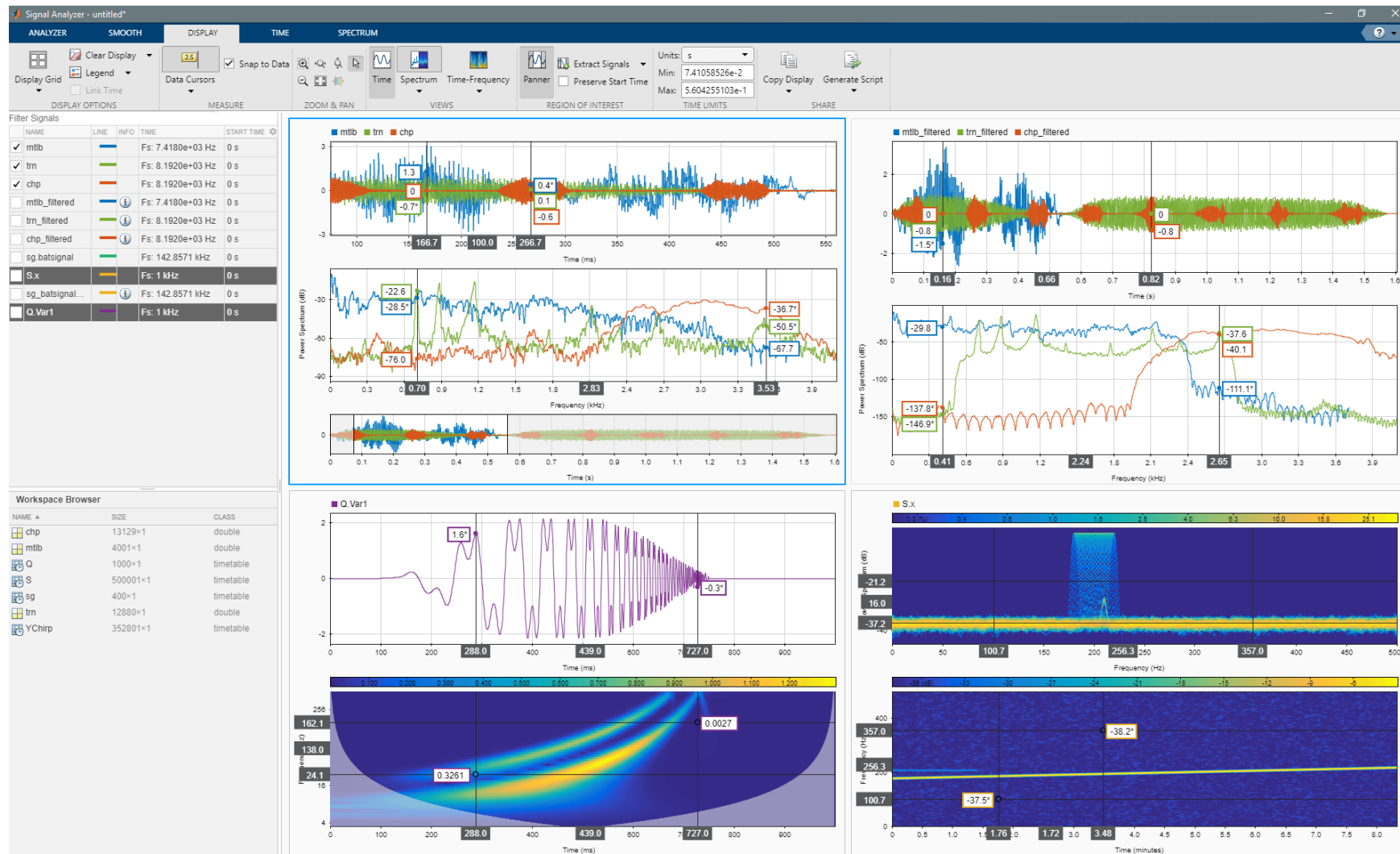
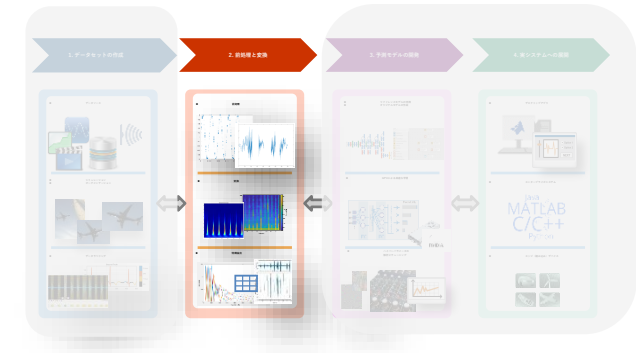
- Spectrum Analyzer

Agenda

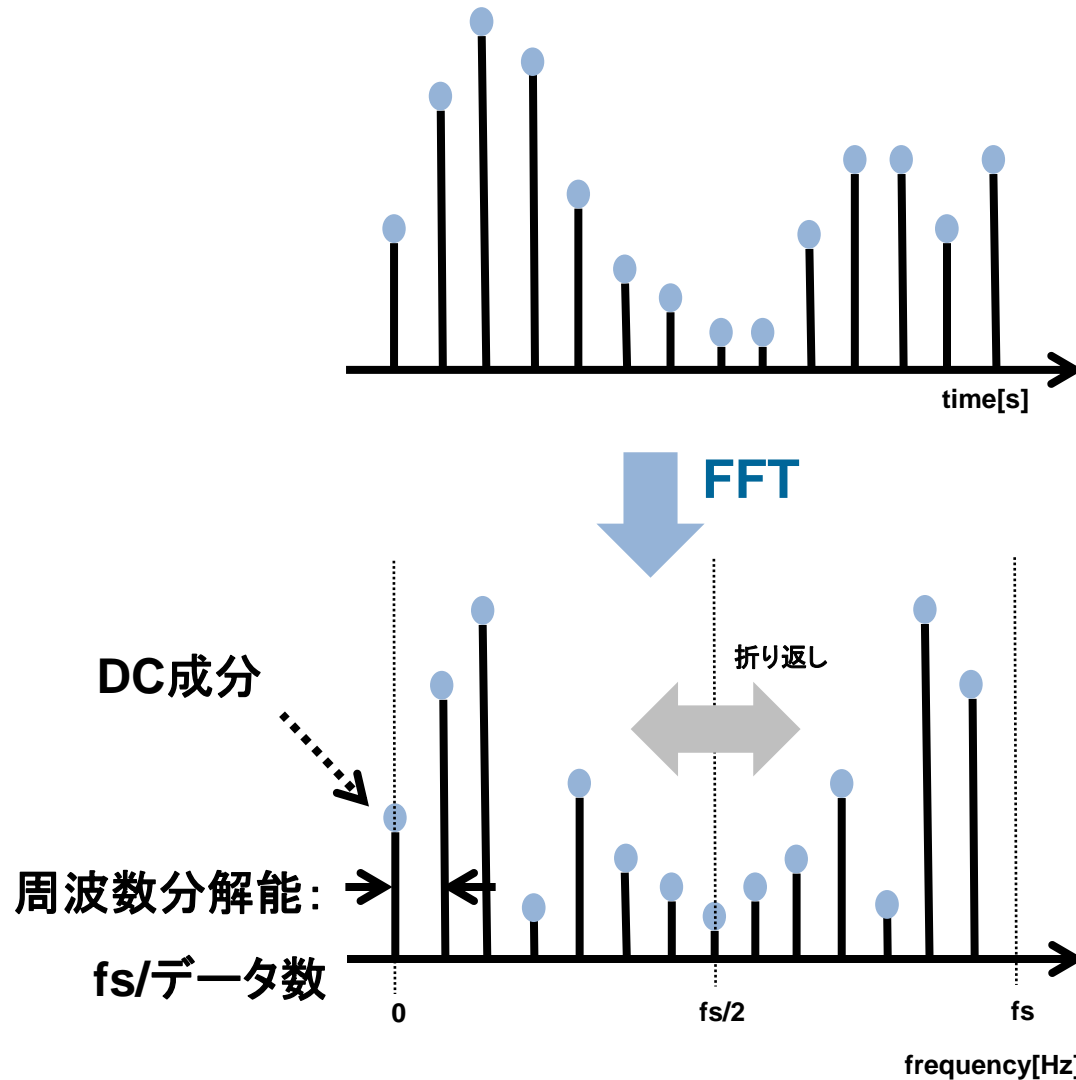
- はじめに
- デモ
 1. データセットの準備：音の録音・再生を楽しみたい
 2. 前処理・特徴抽出：**波形の表示・解析**を楽しみたい
 3. 予測モデルの開発：深層ネットワークの編集を楽しみたい
 4. 実システムへの展開：エッジ開発を楽しみたい
- まとめ

信号アナライザー (>> signalAnalyzer)

時間および周波数軸での特性の可視化・解析



基礎技術：周波数変換（FFT：高速フーリエ変換）



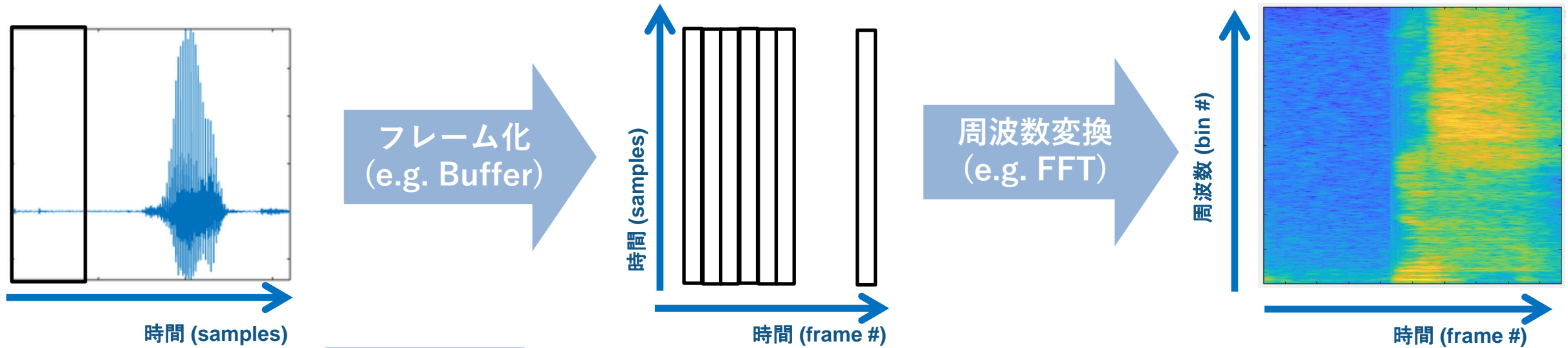
$$FFT : X(k) = \sum_{j=1}^N x(j) \omega_N^{(j-1)(k-1)}$$
$$IFFT : x(j) = (1/N) \sum_{k=1}^N X(k) \omega_N^{-(j-1)(k-1)}$$

where

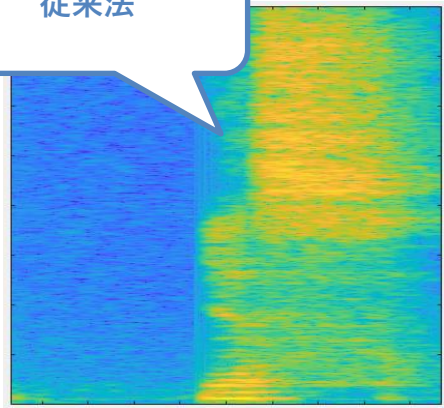
$$\omega_N = e^{(-2\pi i)/N}$$

- データタイプは複素数（振幅と位相の情報を持つ）
- $fs/2$ （ナイキスト）で折り返し
- 変換前と後で同じデータ点数
- データ点数が多いほど、周波数分解能は上がる

基礎技術：時間-周波数変換

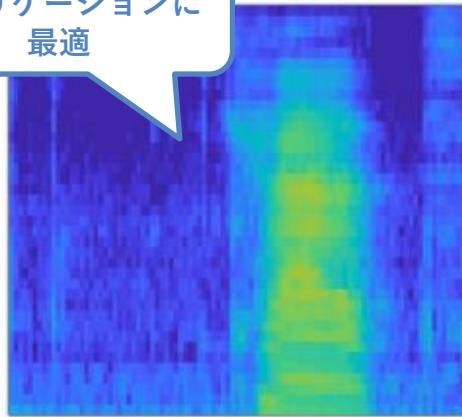


従来法



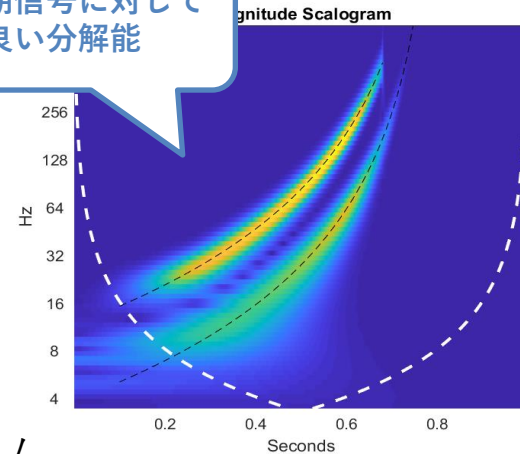
スペクトログラム

音声・音響の
アプリケーションに
最適



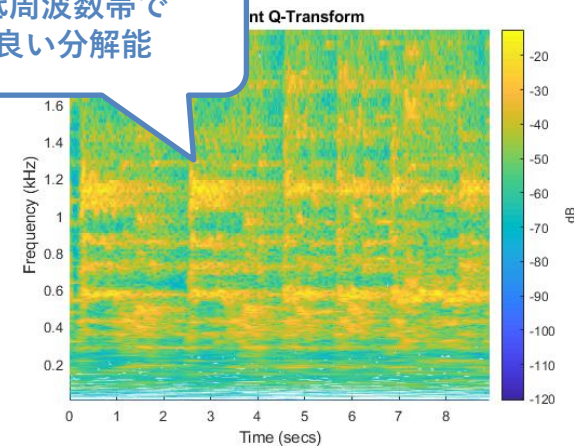
知覚を加味したスペクトログラム
(e.g. Mel, Bark)

非周期信号に対して
良い分解能



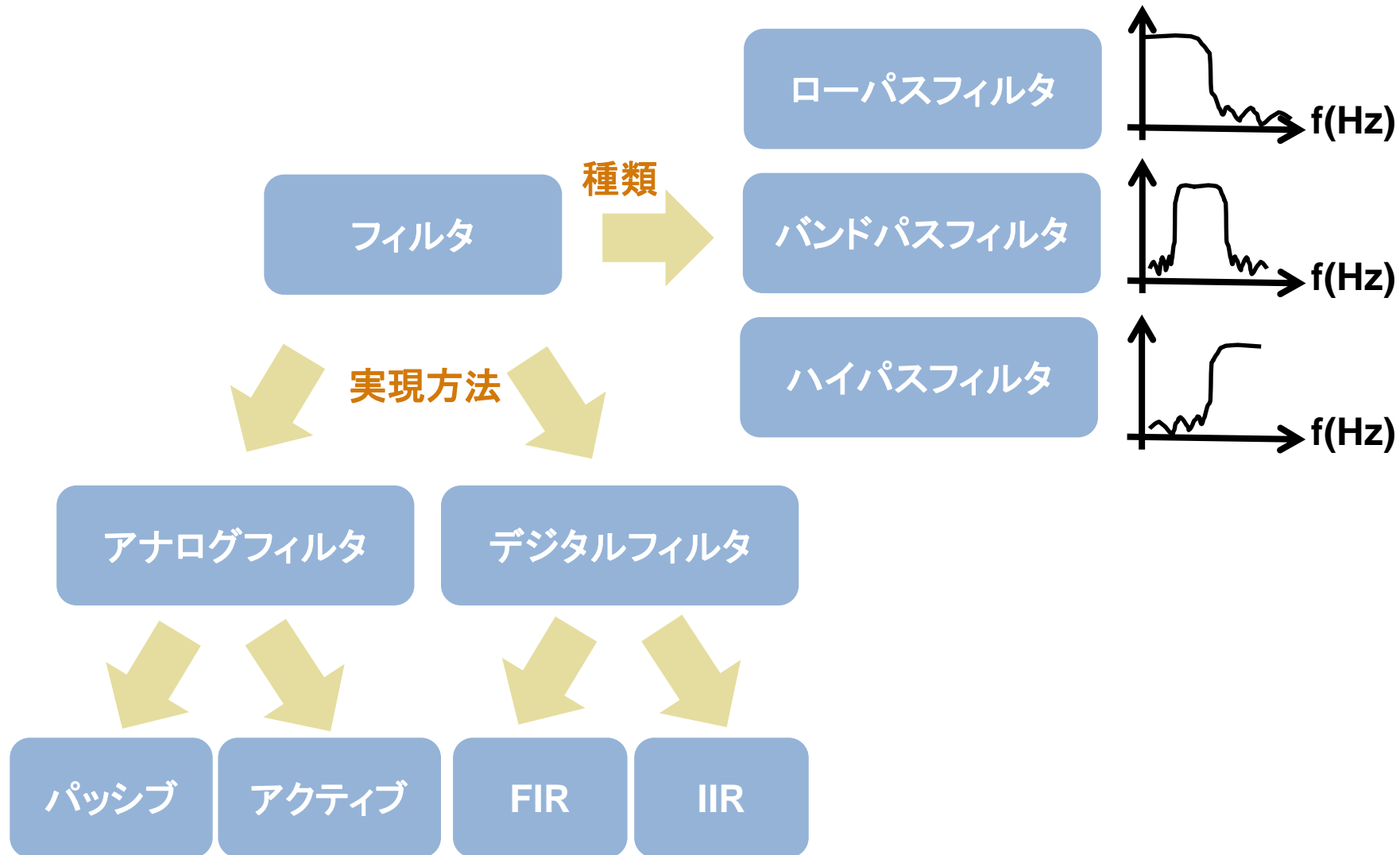
ウェーブレット
スカログラム

低周波数帯で
良い分解能



定Q変換

基礎技術：フィルター処理



特徴抽出のための信号の分割

```
% 特徴抽出オブジェクトの設定
afe = audioFeatureExtractor(...
'SampleRate',fs, ... % サンプリング周波数
'Window',hann(0.025*fs,'periodic'), ... % 窓関数
'OverlapLength',round(0.015*fs), ... % オーバーラップ長
'barkSpectrum',true); % Barkスペクトラム (Appendix参照)

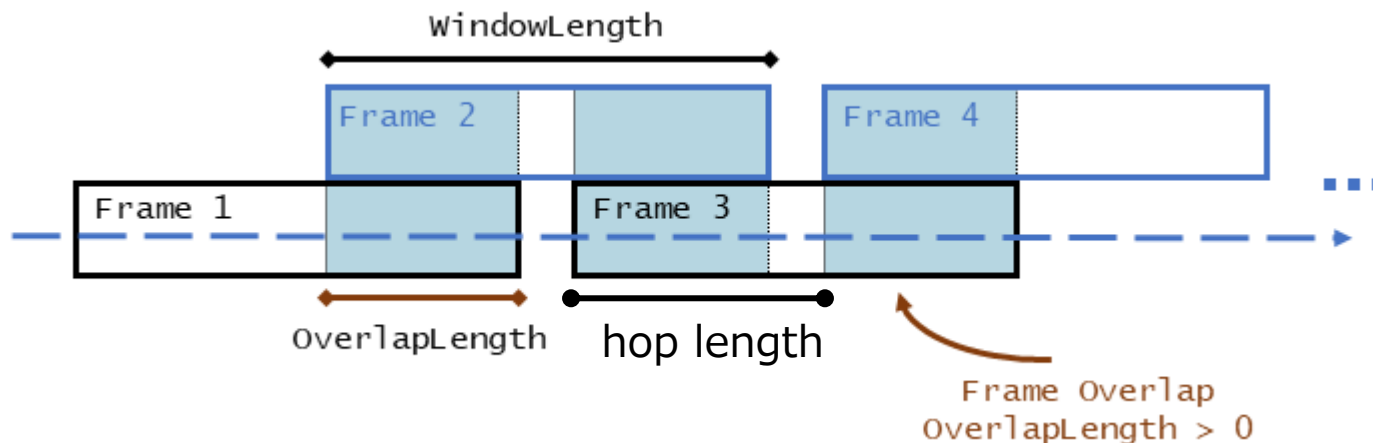
setExtractorParams(afe,'barkSpectrum','NumBands',64);
```

```
% YAMNet(事前学習済みネットワーク)
net = yamnet; % YAMNetの定義
```

```
>> net.Layers(1).InputSize % 入力サイズの確認
```

```
ans =
    96    64     1
```

YAMNetの期待する
入力サイズ (96*64) に合わせる



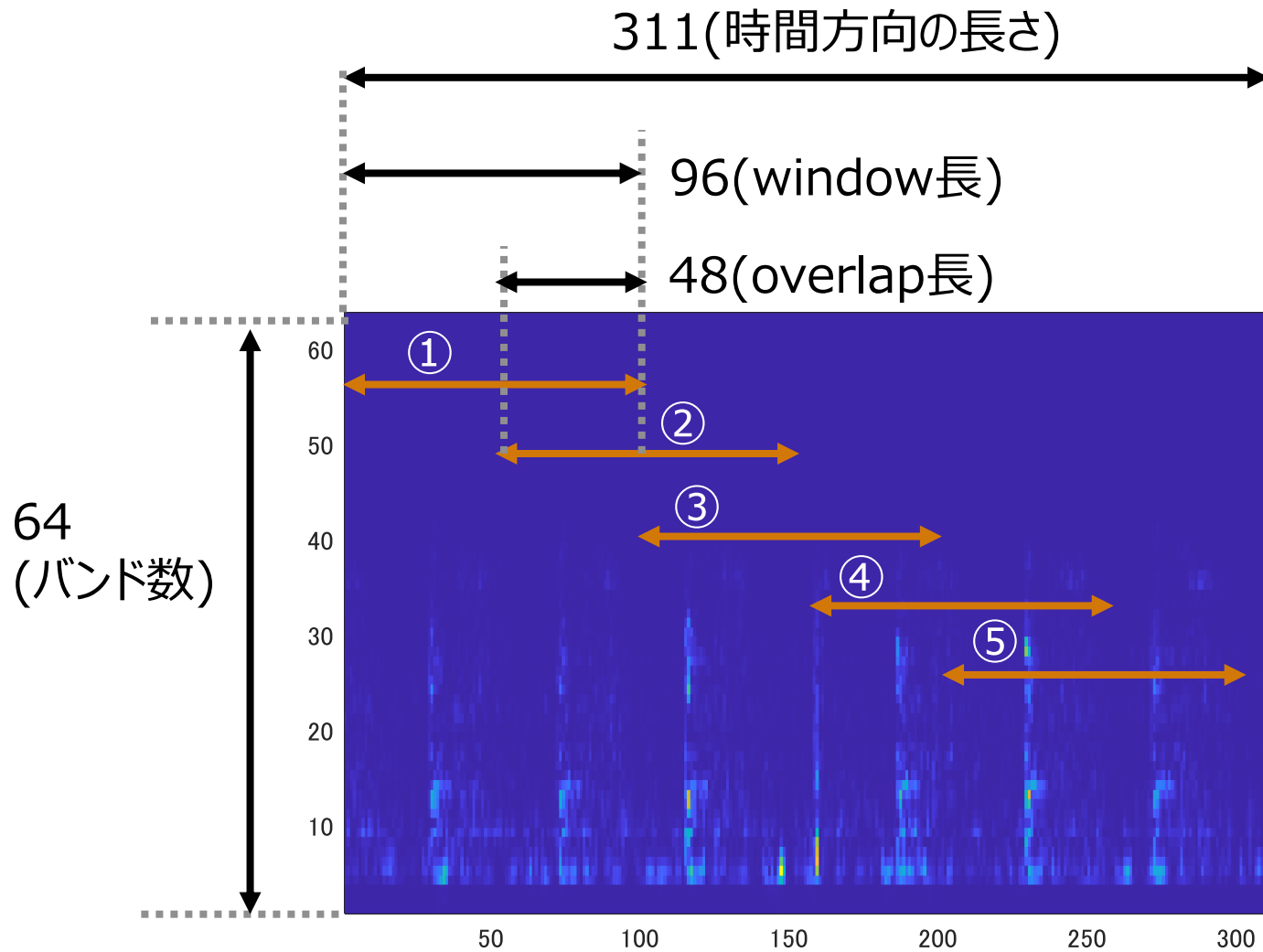
分割フレーム数 = $\text{floor}((n\text{Rows} - \text{winLen}) / \text{hopLen}) + 1$

- $n\text{Rows}$ -- 信号長
- winLen -- 窓関数長
- hopLen -- 始点から次フレーム開始までのサンプル数

e.g.) 本例題の場合

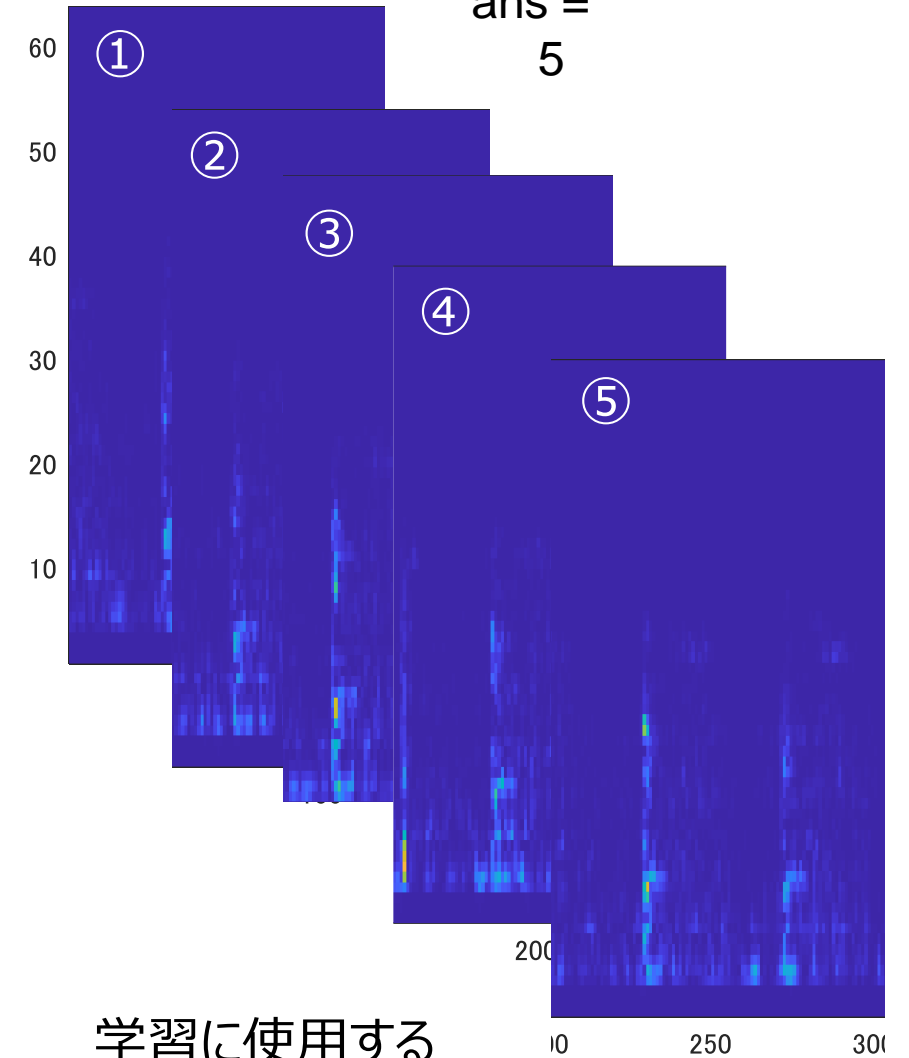
```
>> floor(length(5000)-400)/(400-240)+1
ans =
    311
```


分割のイメージ



exrtact関数で抽出したスペクトログラム(311*64)

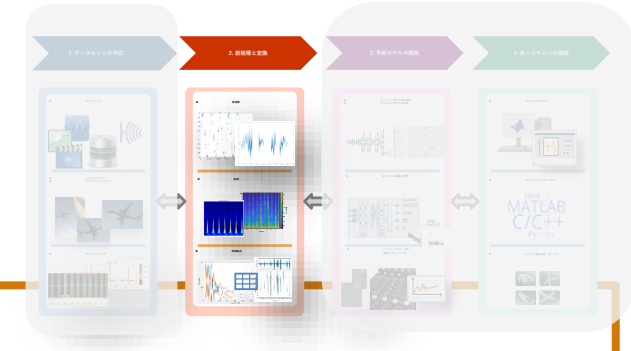
分割数
 $\gg \text{floor}((311-96)/48)+1$
ans =
5



学習に使用する
スペクトログラム(96*64)

YAMNet用特徴抽出 (>> **yamnetPreprocess**)

事前学習済みネットワークのためのスペクトログラム



```

while hasdata(ads)
    [audioIn, fileInfo] = read(ads);
    features = extract(afe, audioIn);
    features = log10(features + single(0.001));
    [numSpectrums, numBands] = size(features);
    numSpectrograms = floor((numSpectrums - numSpectrumsPerSpectrogram)/numSpectrumsHopBetweenSpectrograms) + 1;

    for hop = 1:numSpectrograms
        range = 1 + ...
            numSpectrumsHopBetweenSpectrograms*(hop-1) : numSpectrumsHopBetweenSpectrograms*(hop-1) + numSpectrumsPerSpectrogram;
        trainFeatures = cat(4, trainFeatures, features(range, :));
    end
end

```

```

while hasdata(ads)
    [audioIn, fileInfo] = read(ads);
    features = yamnetPreprocess(audioIn, fs);
    trainFeatures = cat(4, allFeatures, features);
end

```

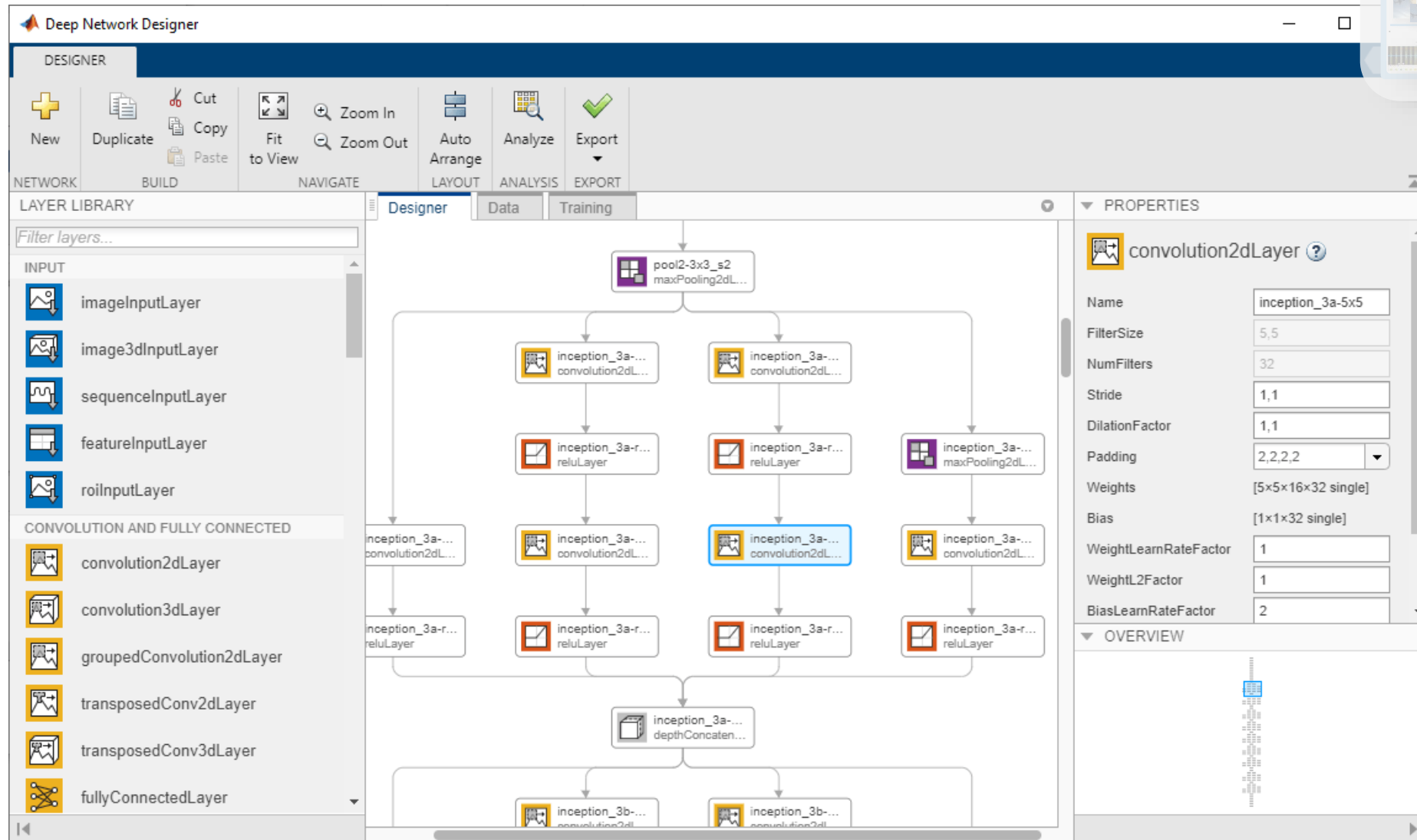
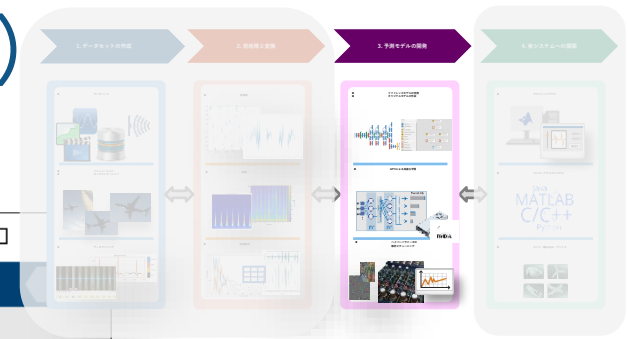
スペクトログラムのサイズ設定や、
部分時系列分割を自動化

Agenda

- はじめに
- デモ
 1. データセットの準備：音の録音・再生を楽しみたい
 2. 前処理・特徴抽出：波形の表示・解析を楽しみたい
 3. 予測モデルの開発：**深層ネットワークの編集**を楽しみたい
 4. 実システムへの展開：エッジ開発を楽しみたい
- まとめ

ディープネットワークデザイナー (>> **deepNetworkDesigner**)

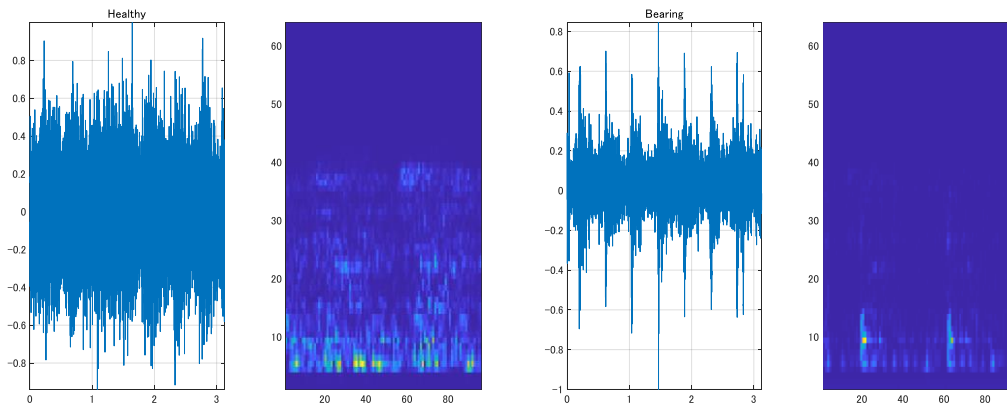
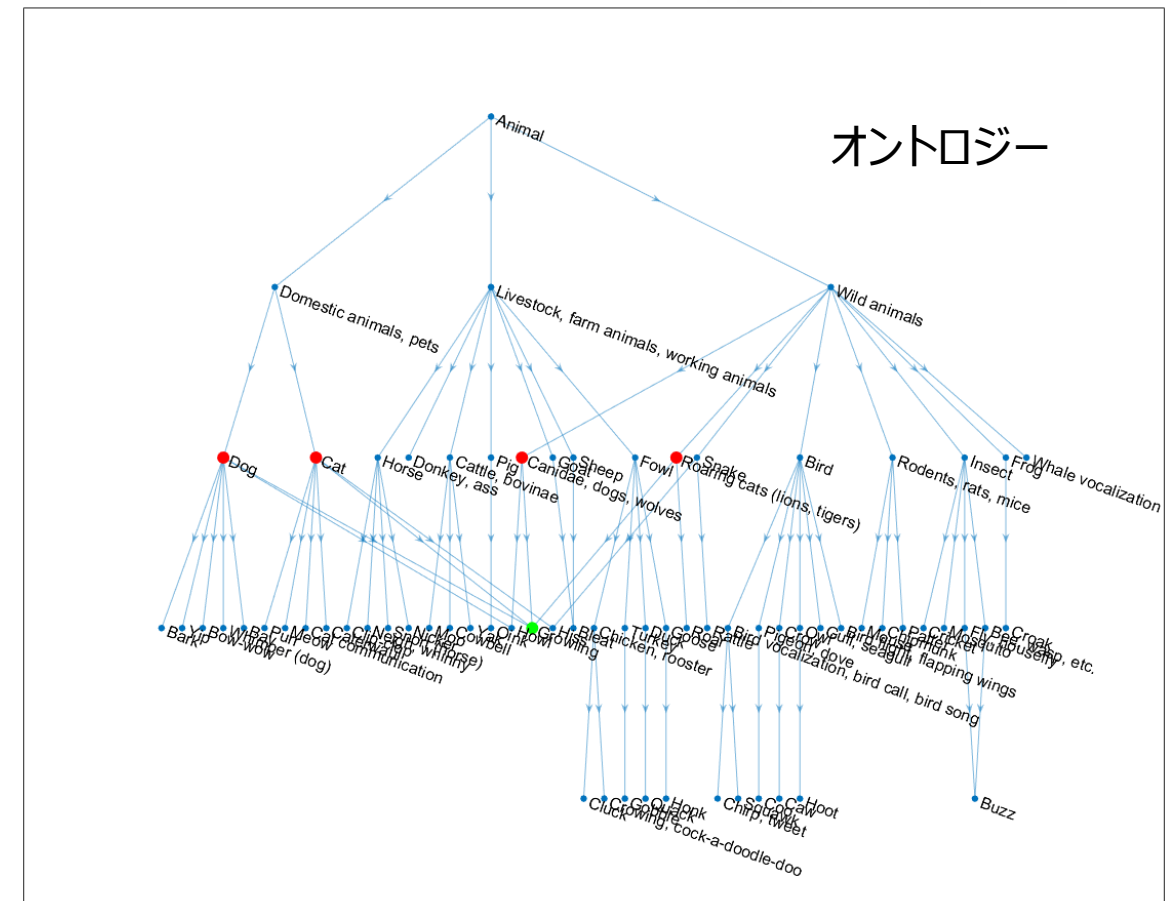
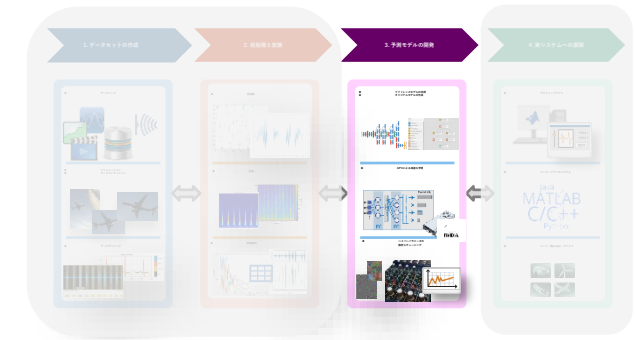
音声・音響信号用事前学習済みネットワーク



YAMNet (>> **yamnet**)

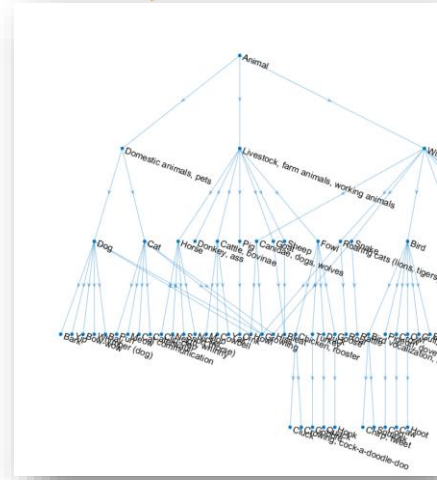
音声・音響信号用事前学習済みネットワーク

- AudioSetを用いた音声・音響分類ネットワークを定義
- 512種類の分類
- オントロジー表示も可能 (>> yamnetGraph)
- 転移学習により独自の音判別機を作成可能



□正常

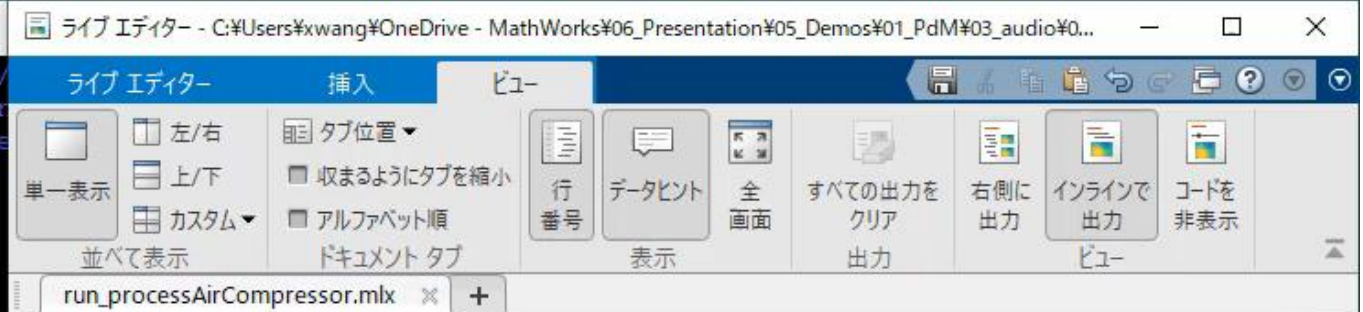
□ベアリング異常



Agenda

- はじめに
- デモ
 1. データセットの準備：音の録音・再生を楽しみたい
 2. 前処理・特徴抽出：波形の表示・解析を楽しみたい
 3. 予測モデルの開発：深層ネットワークの編集を楽しみたい
 4. 実システムへの展開：**エッジ開発**を楽しみたい
- まとめ

```
pi@raspberrypi-mn2M91tFd1: ~/airCompressor_spectrogram/MATLAB_ws/R2021a/C/...  
pi@raspberrypi-mn2M91tFd1:~/airCompressor_spectrogram/  
ws/R2021a/C/Users/xwang/OneDrive_-_MathWorks/06_Presen  
05_Demos/01_PdM/03_audio/01_airCompressor/depolyRaspbe  
S_PdMwebinar_20210521 $
```



Raspberry Pi 上で異音判定アルゴリズムを実行

初期化

```
1 clear; close all;  
2 format short; format compact;  
3 % 乱数群の制御  
4 rng('default');
```

Raspberry Piへの接続

```
5 r = raspi
```

実行可能ファイルのパスを取得

```
6 applicationDirPaths = raspi.utils.getRemoteBuildDirectory('applicationName','prc  
7 targetDirPath = applicationDirPaths{1}.directory;
```

データフォルダ

06_Presentation > 05_Demos > 00_dataset > 03_airCompressor > AirCompressorDataSet > AirCompressorDataSet > 99_csvFiles > Healthy

Healthy001.csv Healthy002.csv Healthy003.csv
Healthy004.csv Healthy005.csv Healthy006.csv

UTF-8

LF

スクリプト

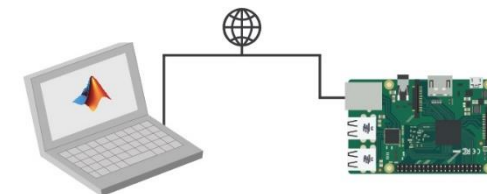
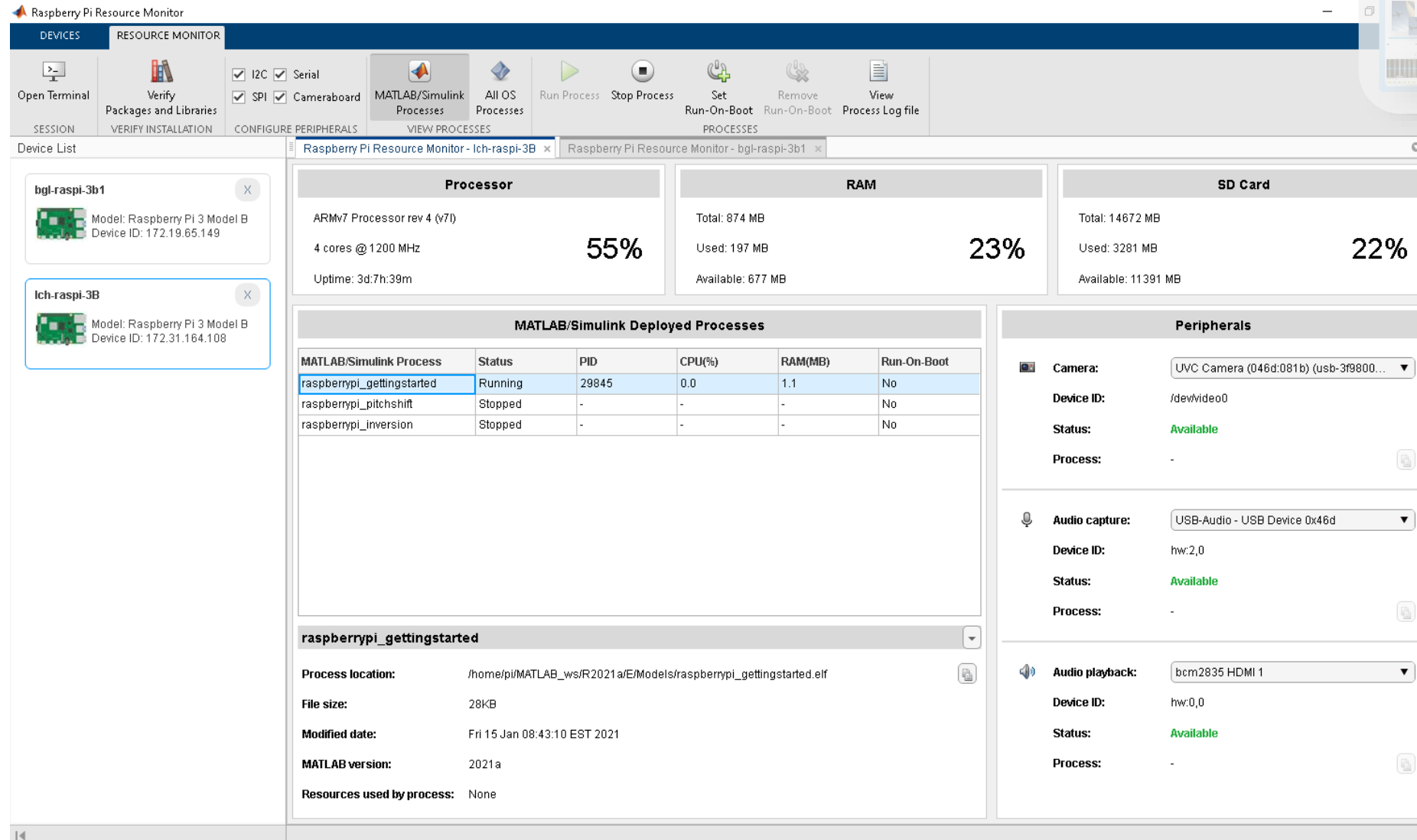
行 4

列 16



Raspberry Pi リソース モニター

(>> **rasberryPiResourceMonitor**) RasPiの状態監視



まとめ：本セミナーでは…

ご紹介したアプリ
ご紹介できなかったアプリ

1. データセットの準備：

Audio Test Benchで音の録音・再生を楽しみました
信号ラベラーでラベリング作業も楽しめます

2. 前処理・特徴抽出：

信号アナライザで波形の表示・解析を楽しみました
フィルターデザイナーでノイズ除去も楽しめます
診断特徴デザイナーで特徴抽出・判別も楽しめます

3. 予測モデルの開発：

ディープネットワークデザイナーで深層ネットワークの編集を楽しみました
分類学習器で機械学習も楽しめます

4. 実システムへの展開：

Raspberry Pi リソースモニターでエッジ開発を楽しみました
アプリケーションコンパイラでアルゴリズムの配布も楽しめます