최신 AI 기반 시스템에서 데이터 세트의 중요성
– 음성 인식 AI

장규환, MathWorks

Deep learning is a key technology driving the AI megatrend

ARTIFICIAL INTELLIGENCE

Any technique that enables machines to mimic human intelligence

MACHINE LEARNING

Statistical methods that enable machines to "learn" tasks from data without explicitly programming

DEEP LEARNING

Neural networks with many layers that learn representations and tasks "directly" from data

1950s

1980s

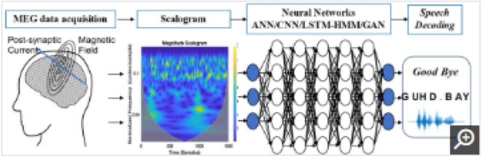
2010s

What does it take to develop an effective real-world deep learning system for signal processing applications?

Deep learning use in signal processing applications is growing rapidly

UT Austin Researchers Convert Brain Signals to Words and Phrases Using Wavelets and Deep Learning

> "MATLAB is an industry-standard tool, and one that you can trust. It is easier to learn than other languages, and its toolboxes help you get started in new areas because you don't have to start from scratch."
>
> — *Dr. Jun Wang, UT Austin*

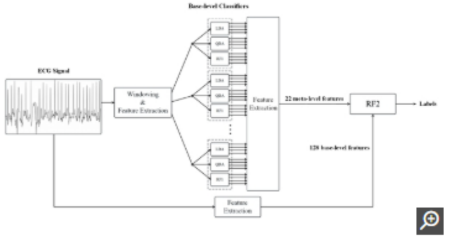


Classifying the brain signals corresponding to the imagined word "goodbye" using feature extraction and deep neural networks.

https://www.mathworks.com/company/user_stories/ut-austin-researchers-convert-brain-signals-to-words-and-phrases-using-wavelets-and-deep-learning.html

MATLAB Based Algorithm Wins the 2017 PhysioNet/CinC Challenge to Automatically Detect Atrial Fibrillation

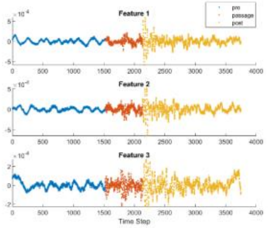
> "I don't think MATLAB has any strong competitors for signal processing and wavelet analysis. When you add in its statistics and machine learning capabilities, it's easy to see why nonprogrammers enjoy using MATLAB, particularly for projects that require combining all these methods."
>
> — *Ali Bahrami Rad, Aalto University*



Block diagram for Black Swan's atrial fibrillation detection algorithm.

https://www.mathworks.com/company/user_stories/matlab-based-algorithm-wins-the-2017-physionet-cinc-challenge-to-automatically-detect-atrial-fibrillation.html

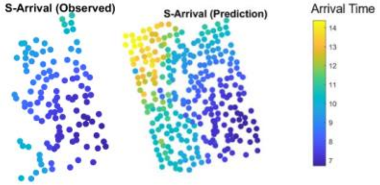
Shell performs Seismic Event Detection with Deep Learning



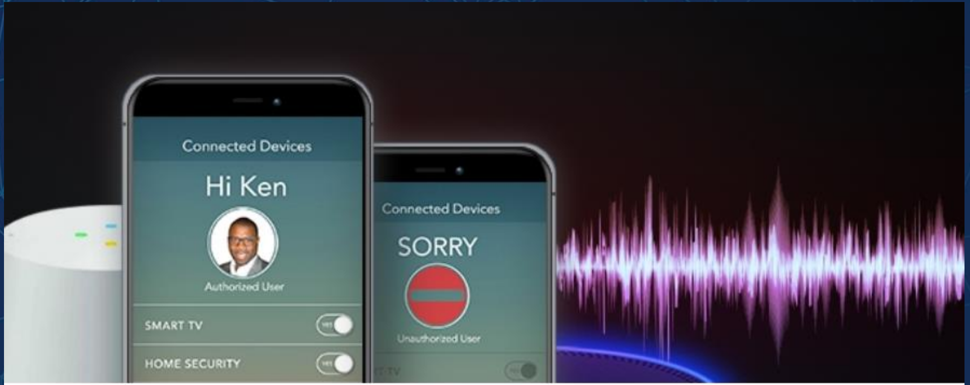
Challenges
- Terabytes of passive seismic data from geophones
- Traditional methods time/labor intensive (5 months &~ $100K)
- Event detection inconsistent/unreliable in 'low' signal to noise records

Solution
- Train LSTM network to detect P-wave and S-wave arrivals via sequence-to-sequence classification

Results
- >98% accuracy for arrival prediction
- Networks generalizes to other data (sites, source mechanisms)



https://library.seg.org/doi/pdf/10.1190/segam2019-3215081.1



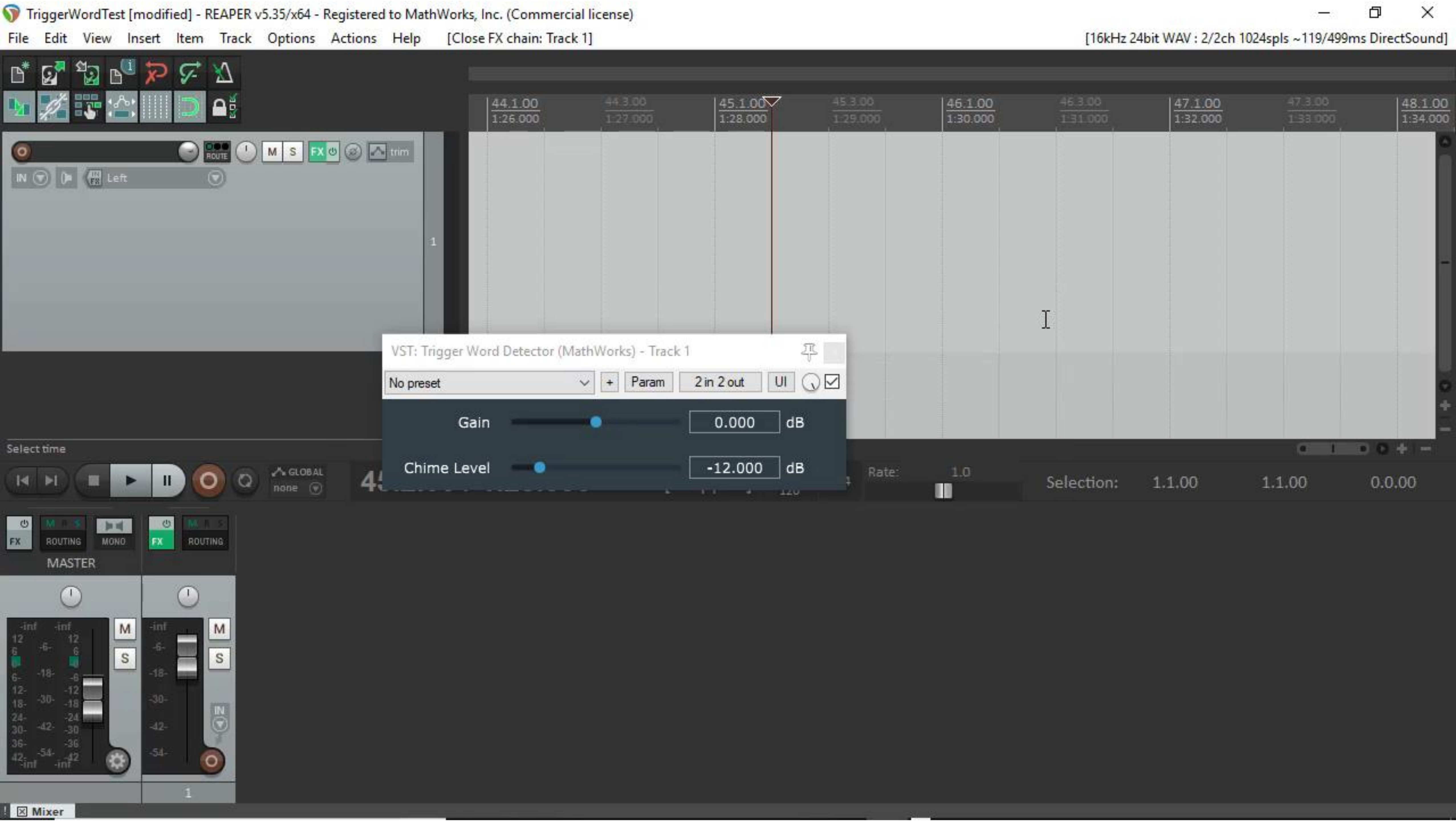
Voice Interface: The Touchscreen of the Next Century

How AI and Signal Processing Came Together to Track the DNA of Sound

https://www.mathworks.com/company/mathworks-stories/ai-signal-processing-for-voice-assistants.html

A Practical Example: Trigger Word Detection

(The embedded gateway to your cloud-based voice assistant)


TriggerWordTest [modified] - REAPER v5.35/x64 - Registered to MathWorks, Inc. (Commercial license)
File Edit View Insert Item Track Options Actions Help [Close FX chain: Track 1]
[16kHz 24bit WAV : 2/2ch 1024spls ~119/499ms DirectSound]
44.1.00 1:26.000
45.1.00 1:28.000
46.1.00 1:30.000
47.1.00 1:32.000
48.1.00 1:34.000
VST: Trigger Word Detector (MathWorks) - Track 1
No preset
Param
2 in 2 out
UI
Gain
0.000
dB
Chime Level
-12.000
dB
Select time
Rate:
1.0
Selection:
1.1.00
1.1.00
0.0.00
MASTER
ROUTING
MONO
Mixer

Find most of the code for this example online

MathWorks® Products Solutions Academia Support Community Events

Help Center

Search Support

CONTENTS

Documentation Examples Functions Blocks Apps Videos Answers

Keyword Spotting in Noise Using MFCC and LSTM Networks

This example shows how to identify a keyword in noisy speech using a deep learning network. In particular, the example uses a Bidirectional Long Short-Term Memory (BiLSTM) network ar (MFCC).

Introduction

Keyword spotting (KWS) is an essential component of voice-assist technologies, where the user speaks a predefined keyword to wake-up a system before speaking a complete command c

This example trains a KWS deep network with feature sequences of mel-frequency cepstral coefficients (MFCC). The example also demonstrates how network accuracy in a noisy environn augmentation.

This example uses long short-term memory (LSTM) networks, which are a type of recurrent neural network (RNN) well-suited to study sequence and time-series data. An LSTM network ca (`lstmLayer`) can look at the time sequence in the forward direction, while a bidirectional LSTM layer (`bilstmLayer`) can look at the time sequence in both forward and backward directions.

The example uses the google Speech Commands Dataset to train the deep learning model. To run the example, you must first download the data set. If you do not want to download the da in MATLAB® and typing `load("KWSNet.mat")` at the command line.

Example Summary

The example goes through the following steps:

1. Inspect a "gold standard" keyword spotting baseline on a validation signal.
2. Create training utterances from a noise-free dataset.
3. Train a keyword spotting LSTM network using MFCC sequences extracted from those utterances.
4. Check the network accuracy by comparing the validation baseline to the output of the network when applied to the validation signal.

https://www.mathworks.com/help/audio/examples/keyword-spotting-in-noise-using-mfcc-and-lstm-networks.html

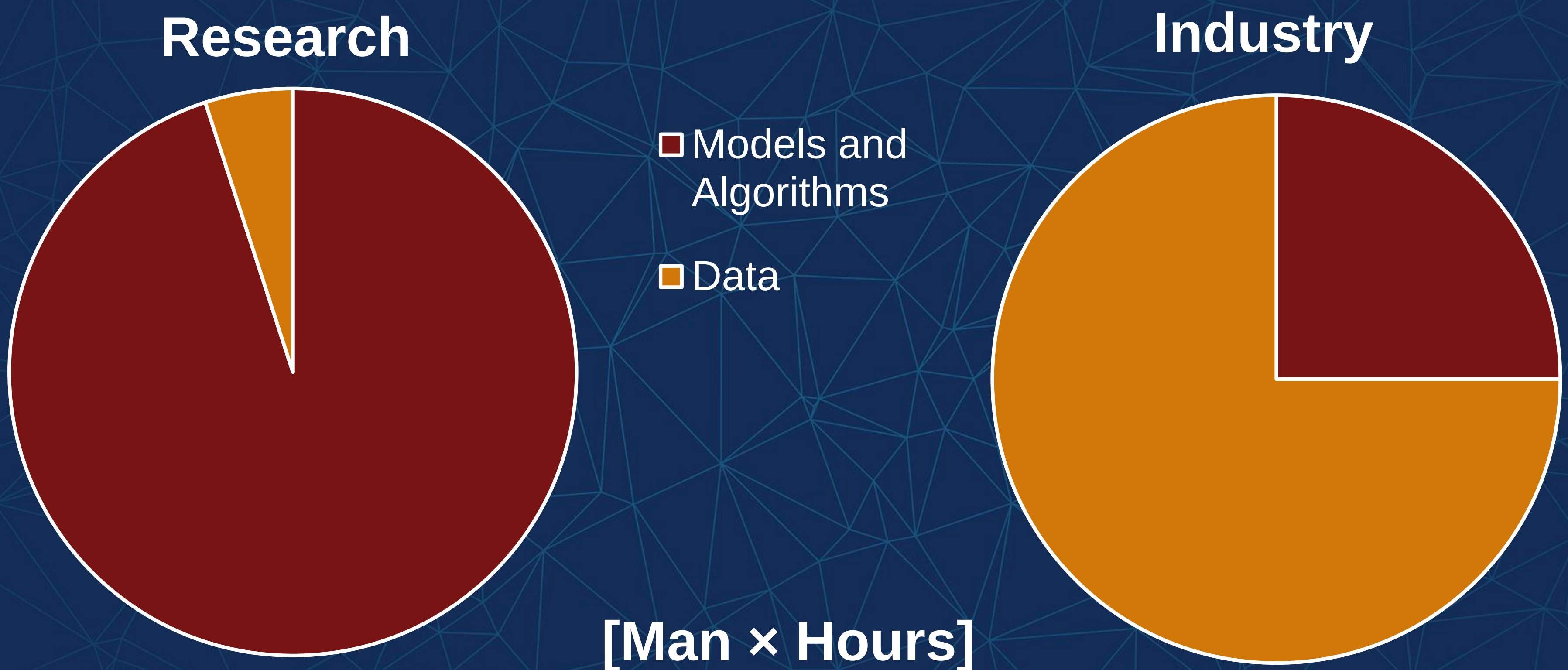
What does it take to develop an effective real-world deep learning system for signal processing applications?

A: "The right deep network design"

"A BiLSTM network with layers of 150 hidden units each, followed by one fully-connected layer and a softmax layer"

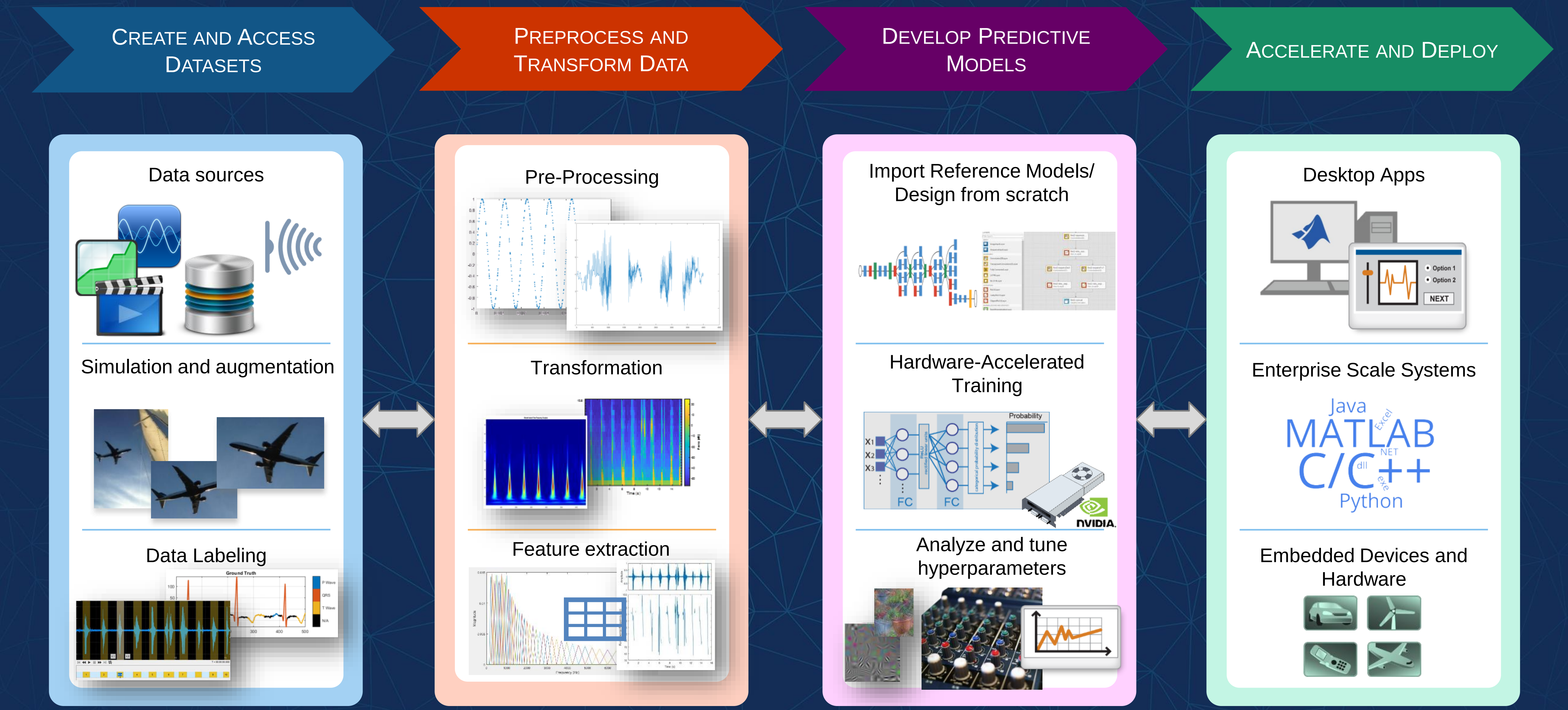
A: "A lot of data, a good dose of signal processing expertise, and the right tools for the specific application in hand"

Data Investments in Deep Learning
Research vs. Industry



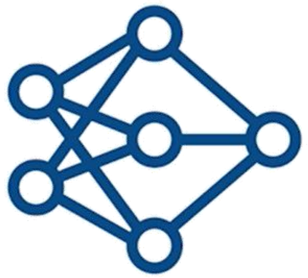
[Man × Hours]

Based on: *Andrej Karpathy* – *Building the Software 2.0 Stack (Spark+AI Summit 2018)*

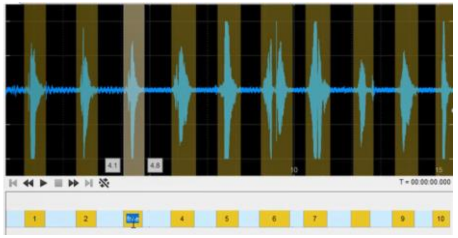


Create and Access Datasets
Preprocess and Transform Data
Develop Predictive Models
Accelerate and Deploy
Data sources
Simulation and augmentation
Data Labeling
Ground Truth
Pre-Processing
Transformation
Feature extraction
Import Reference Models/ Design from scratch
Hardware-Accelerated Training
Probability
FC
FC
NVIDIA.
Analyze and tune hyperparameters
Desktop Apps
Option 1
Option 2
NEXT
Enterprise Scale Systems
Java
Excel
MATLAB
.NET
C/C++
dll
exe
Python
Embedded Devices and Hardware

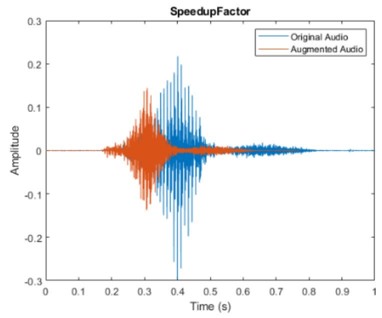
Agenda



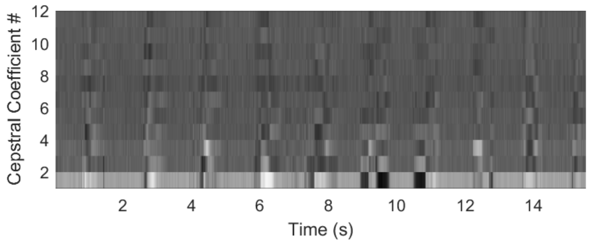
- Basics on training deep neural networks for signals



- Annotating data to train networks for practical applications



- Generating new data – synthesis and augmentation

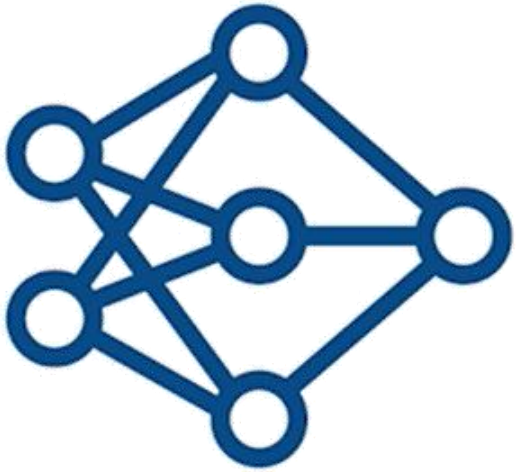


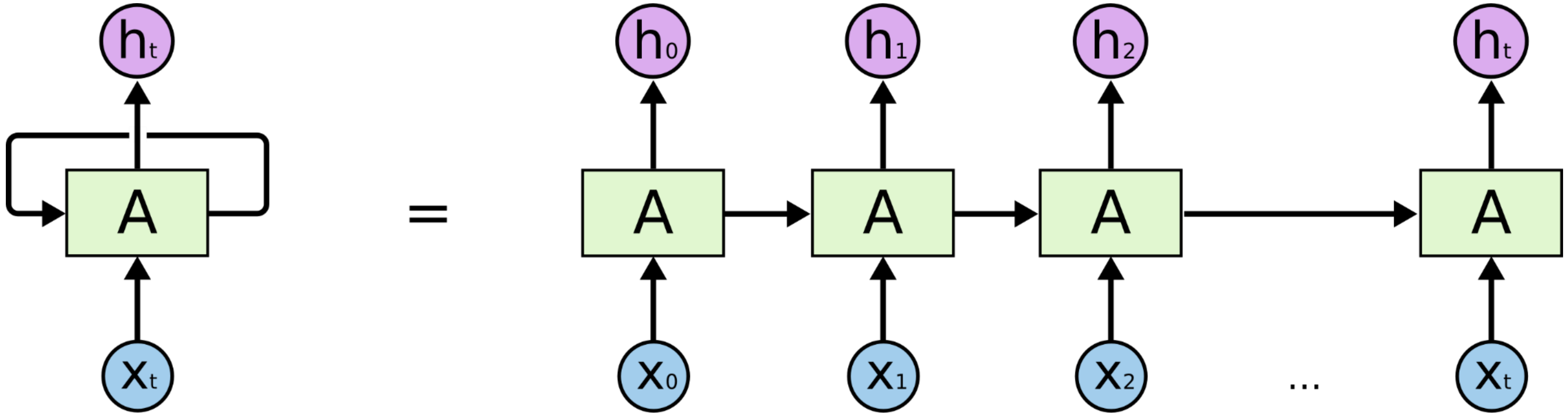
- Creating inputs for deep networks

- From system models to real-time prototypes

Defining a deep network architecture

```
layers = [ ...
    sequenceInputLayer(numFeatures)
    bilstmLayer(150,"OutputMode","sequence")
    bilstmLayer(150,"OutputMode","sequence")
    fullyConnectedLayer(2)
    softmaxLayer
    classificationLayer
    ];
```



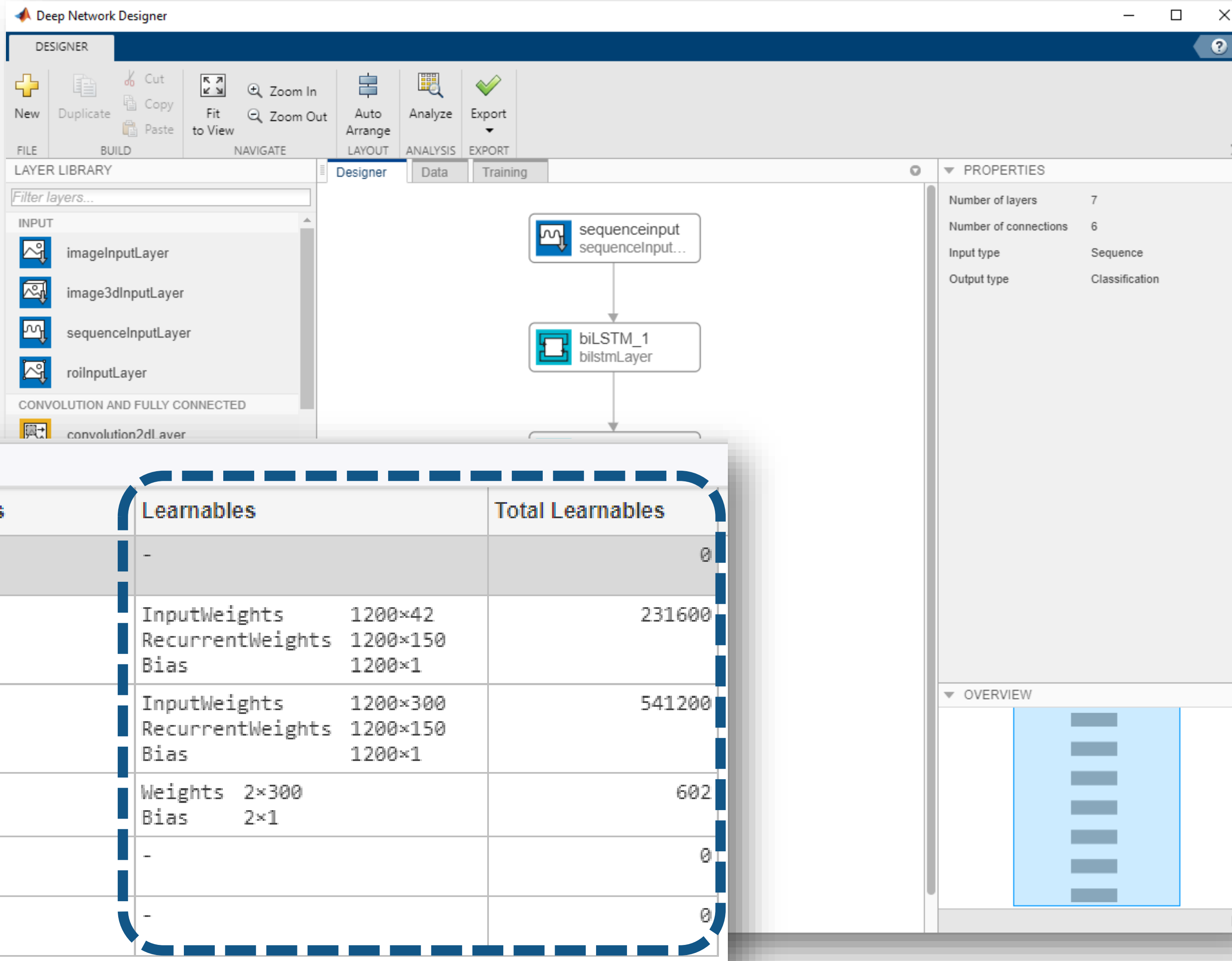


Long Short Term Memory (**LSTM**) Layer

(Recursive Neural Networks, **RNN**)

Defining a deep network architecture

```
layers = [ ...
    sequenceInputLayer(numFeatures)
    bilstmLayer(150,"OutputMode","sequence")
    bilstmLayer(150,"OutputMode","sequence")
    fullyConnectedLayer(2)
    softmaxLayer
    classificationLayer
    ];
```



ANALYSIS RESULT

	Name	Type	Activations	Learnables	Total Learnables
1	sequenceinput Sequence input with 42 dimensions	Sequence Input	42	-	0
2	biLSTM_1 BiLSTM with 150 hidden units	BiLSTM	300	InputWeights 1200×42 RecurrentWeights 1200×150 Bias 1200×1	231600
3	biLSTM_2 BiLSTM with 150 hidden units	BiLSTM	300	InputWeights 1200×300 RecurrentWeights 1200×150 Bias 1200×1	541200
4	fc 2 fully connected layer	Fully Connected	2	Weights 2×300 Bias 2×1	602
5	softmax softmax	Softmax	2	-	0
6	classoutput crossentropyex	Classification Output	-	-	0

Start from published recipes...

Long Short-Term Memory Recurrent Neural Network Architectures for Large Scale Acoustic Modeling

Haşim Sak, Andrew Senior, Françoise Beaufays

Google, USA

Long short-term memory for speaker generalization in supervised speech separation

Jitong Chen[a] and DeLiang Wang[b]

Department of Computer Science and Engineering, The Ohio State University, Columbus, Ohio 43210, USA

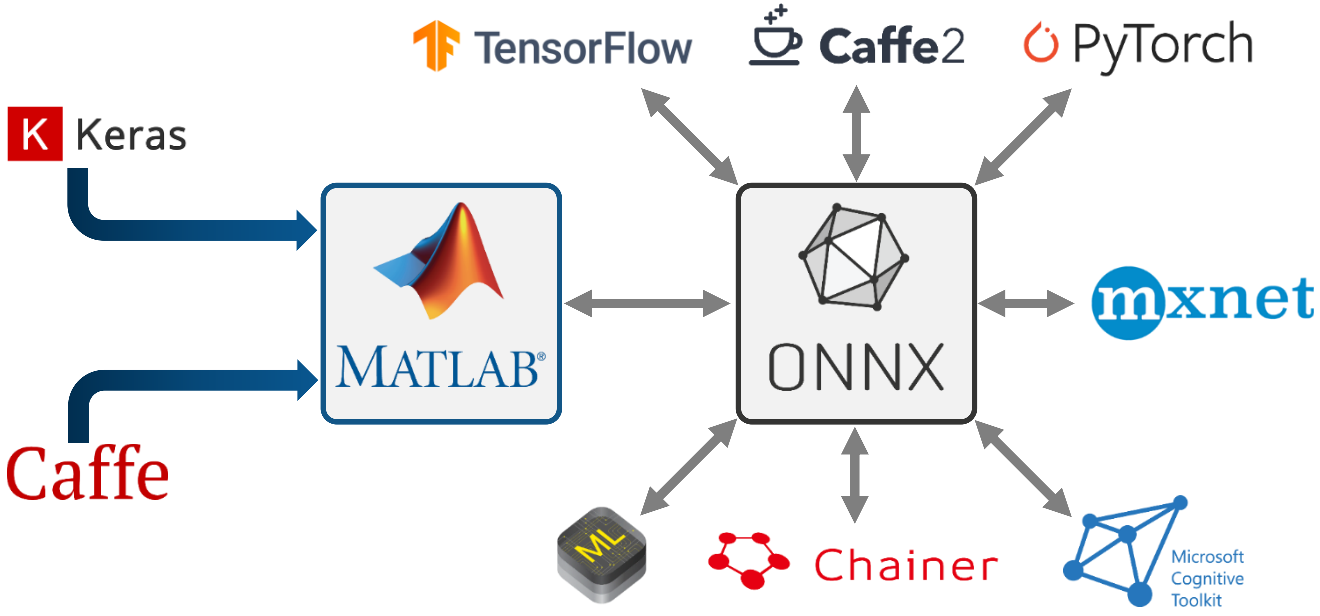
nline 23

An Improved Residual LSTM Architecture for Acoustic Modeling

Lu Huang Department of Electronic Engineering Tsinghua University Beijing, China e-mail: huanglu.th@gmail.com	Jiasong Sun Department of Electronic Engineering Tsinghua University Beijing, China e-mail: sunjiasong@tsinghua.edu.cn
Ji Xu b of Speech Acoustics & Content Understanding ute of Acoustics, Chinese Academy of Sciences	Yi Yang Department of Electronic Engineering Tsinghua University

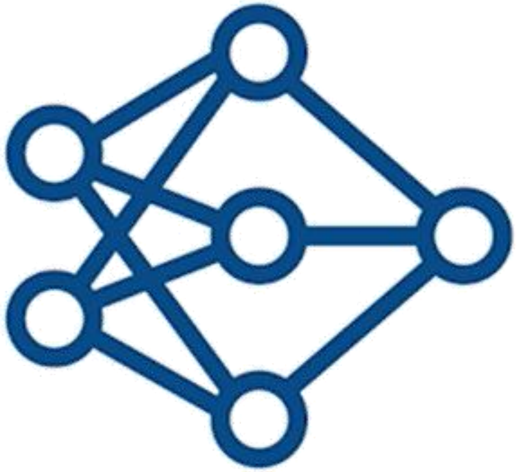
* Random examples found via web search
(No endorsement implied)

...or import models developed by others (including from different frameworks)



Training a deep network

```
layers = [ ...
    sequenceInputLayer(numFeatures)
    bilstmLayer(150,"OutputMode","sequence")
    bilstmLayer(150,"OutputMode","sequence")
    fullyConnectedLayer(2)
    softmaxLayer
    classificationLayer
    ];

maxEpochs     = 10;
miniBatchSize = 64;
options = trainingOptions("adam", ...
    "InitialLearnRate",1e-4,...
    "MaxEpochs",maxEpochs, ...
    "MiniBatchSize",miniBatchSize, ...
    "Shuffle","every-epoch",...
    "Verbose",false, ...
    "ValidationFrequency",floor(numel(TrainingFeatures)/miniBatchSize),...
    "ValidationData",{FeaturesValidationClean.',BaselineV},...
    "Plots","training-progress",...
    "LearnRateSchedule","piecewise",...
    "LearnRateDropFactor",0.1, ...
    "LearnRateDropPeriod",5);

[net,info] = trainNetwork(TrainingFeatures,TrainingMasks,layers,options);
```



b88a9ffecebd:1 () - TigerVNC

MATLAB R2019b

HOME PLOTS APPS EDITOR PUBLISH VIEW

Search Documentation Andy

/ home matlab Documents AudioWebinar Code

Workspace

Name	Value
expectedNumPartitions	128
klstm	4
kovlp	4
loadFeatures	1
LSTMSize	150
LSTMSizes	[75,100,125,150]
LSTMSIzes	[75,100,125,150]
M	1x4 cell
net	1x1 SeriesNetwork
netLayers	6x1 Layer

Current Folder Command Window

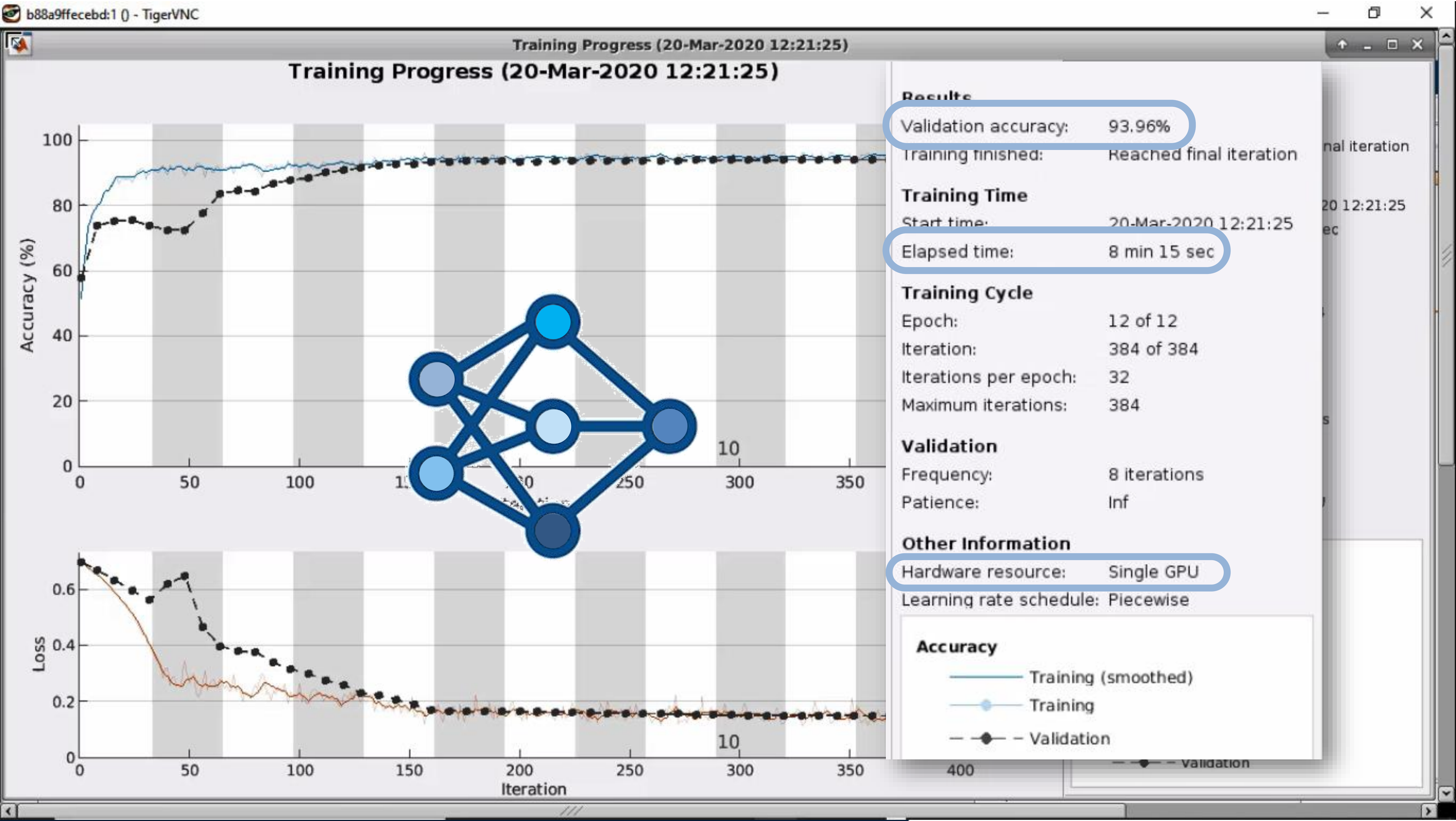
>>

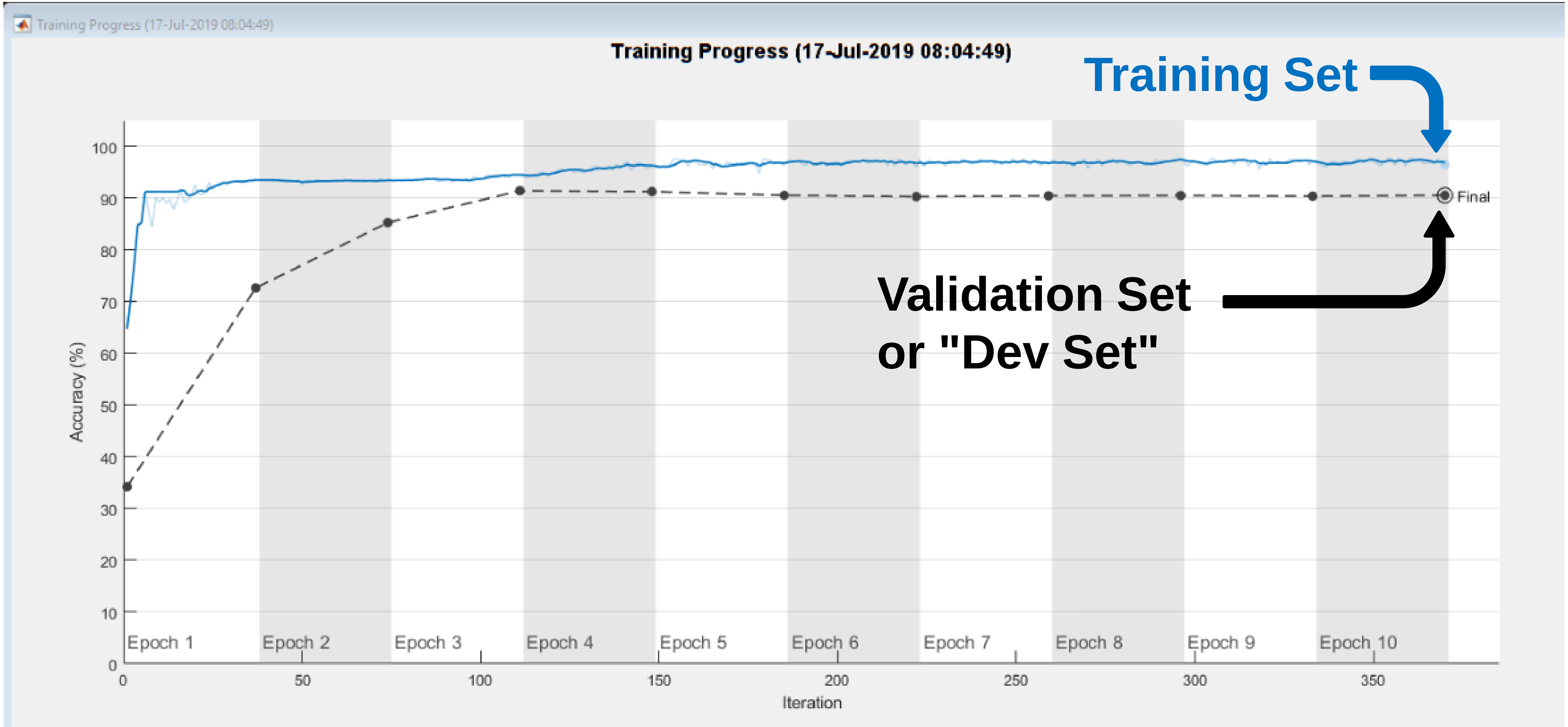
Editor - TrainSingleNetwork.m Variables - feat

TrainSingleNetwork.m

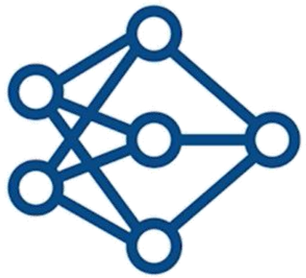
```matlab
netLayers = [ ...
    sequenceInputLayer(numFeatures)
    bilstmLayer(LSTMSizes(klstm),"OutputMode","sequence")
    bilstmLayer(LSTMSizes(klstm),"OutputMode","sequence")
    fullyConnectedLayer(2)
    softmaxLayer
    classificationLayer
    ];

trainOptions = trainingOptions("adam", ...
    "InitialLearnRate",1e-4, ...
    "MaxEpochs",12, ...
    "MiniBatchSize",4, ...
    "Shuffle","every-epoch", ...
    "Verbose",false, ...
    "ValidationFrequency",8, ...
    "ValidationData",{ValidationFeatures{kovlp},ValidationMasks{kovlp}}, ...
    "Plots","training-progress", ...
    "LearnRateSchedule","piecewise", ...
    "LearnRateDropFactor",0.1, ...
    "LearnRateDropPeriod",5,...
    "SequenceLength","Shortest");

%% Network training

tic;
net = trainNetwork(trainingFeatures,trainingMasks,netLayers,trainOptions);
fprintf('Training the network took %g s\n',toc);
```

Busy script Ln 63 Col 1

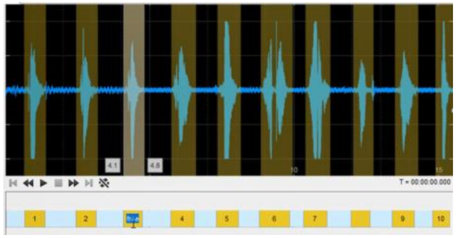

b88a9ffecebd:1 () - TigerVNC
Training Progress (20-Mar-2020 12:21:25)
Training Progress (20-Mar-2020 12:21:25)
Accuracy (%)
Loss
Iteration
Results
Validation accuracy: 93.96%
Training finished: Reached final iteration
Training Time
Start time: 20-Mar-2020 12:21:25
Elapsed time: 8 min 15 sec
Training Cycle
Epoch: 12 of 12
Iteration: 384 of 384
Iterations per epoch: 32
Maximum iterations: 384
Validation
Frequency: 8 iterations
Patience: Inf
Other Information
Hardware resource: Single GPU
Learning rate schedule: Piecewise
Accuracy
Training (smoothed)
Training
Validation

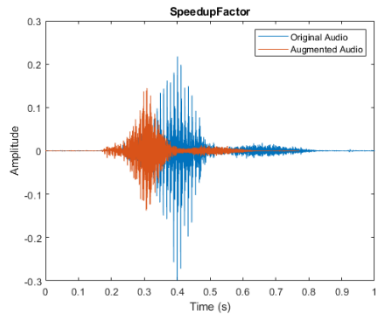

Training Progress (17-Jul-2019 08:04:49)
Training Progress (17-Jul-2019 08:04:49)
Training Set
Validation Set
or "Dev Set"
Final
Accuracy (%)
100
90
80
70
60
50
40
30
20
10
0
Epoch 1
Epoch 2
Epoch 3
Epoch 4
Epoch 5
Epoch 6
Epoch 7
Epoch 8
Epoch 9
Epoch 10
0
50
100
150
200
250
300
350
Iteration

Agenda

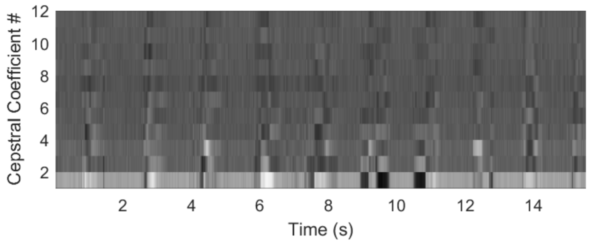


- Basics on training deep neural networks for signals
- Annotating data to train networks for practical applications





- Generating new data – synthesis and augmentation



- Creating inputs for deep networks
- From system models to real-time prototypes

Training, Validation, and Test Data

Your full dataset (All of your **data + labels**)

Training, Validation, and Test Data

GB – TB +

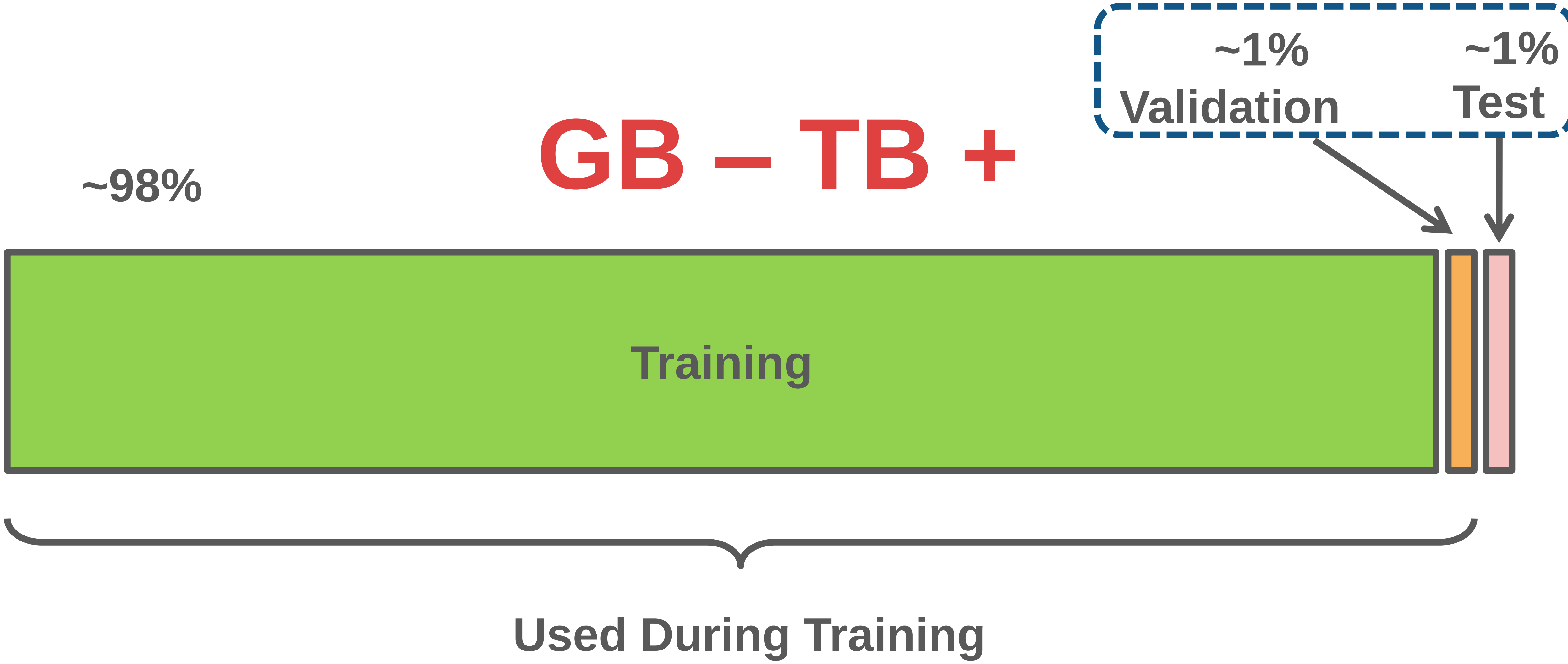
~60% KB – MB ~20% ~20%

Training	Validation	Test

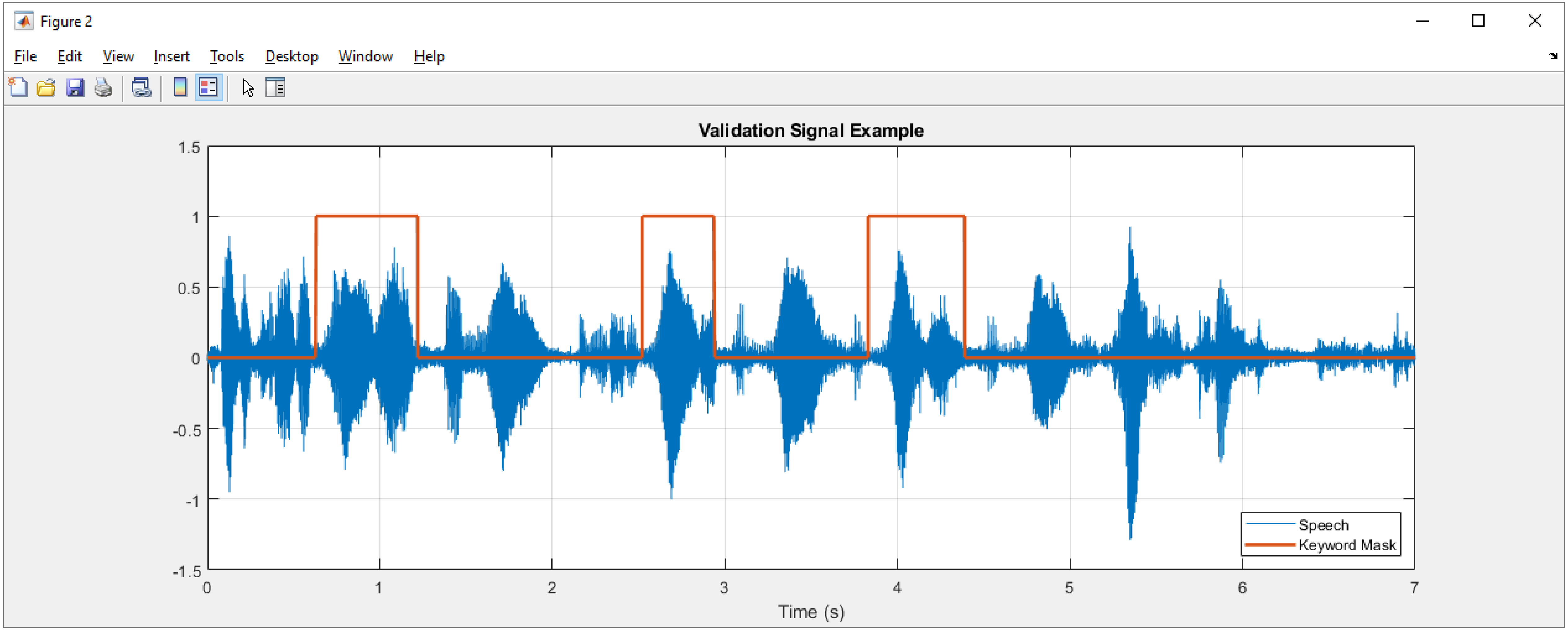
$\underbrace{\text{Training} \quad \text{Validation}}_{\text{Used During Training}}$

Used During Training

Training, Validation, and Test Data



A good validation data sample – Realistic recording, accurately labeled

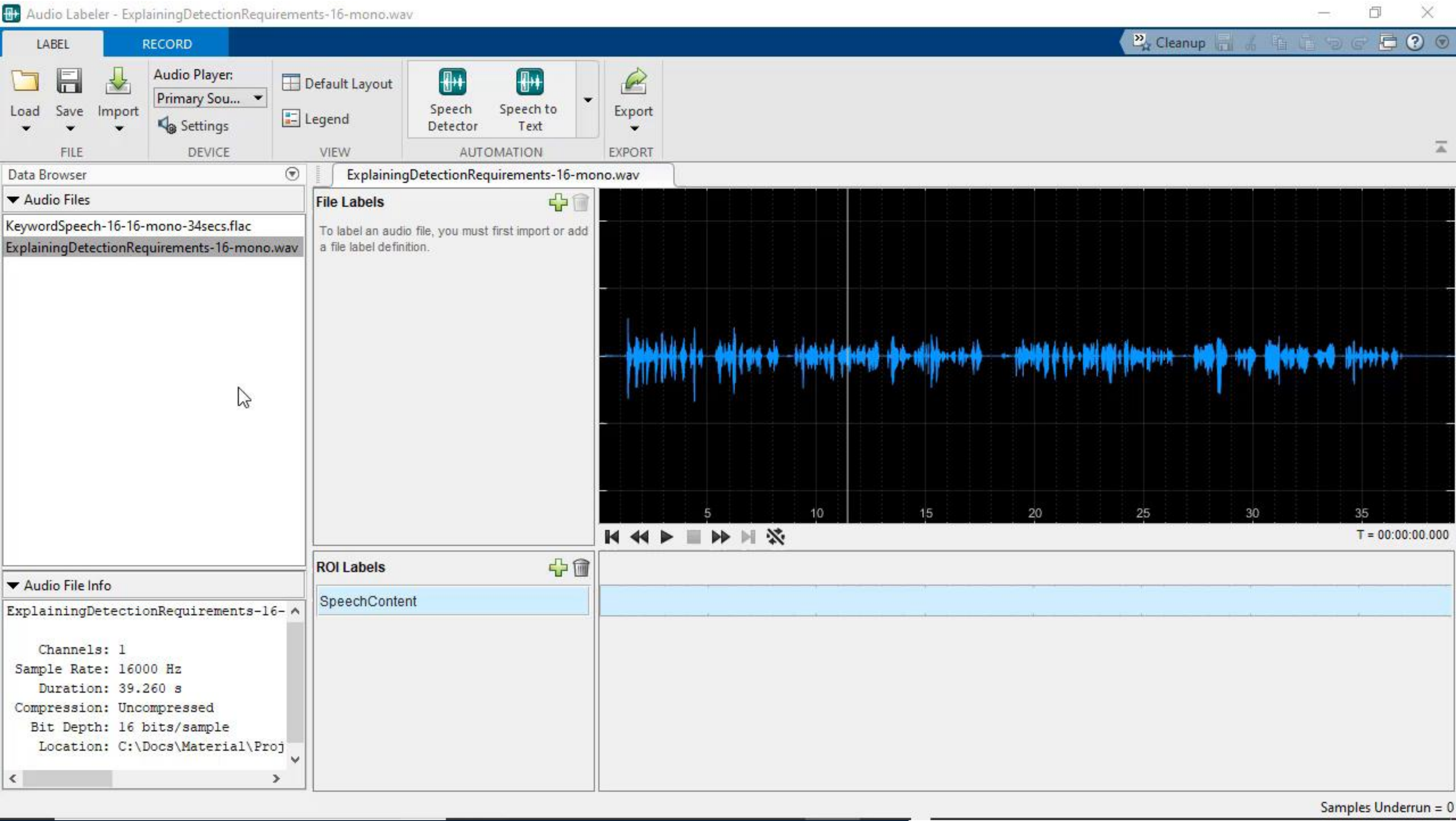


How to label new non-annotated data?

Use an intelligent system trained to carry out a similar tasks with proven accuracy!

For example:

- Humans

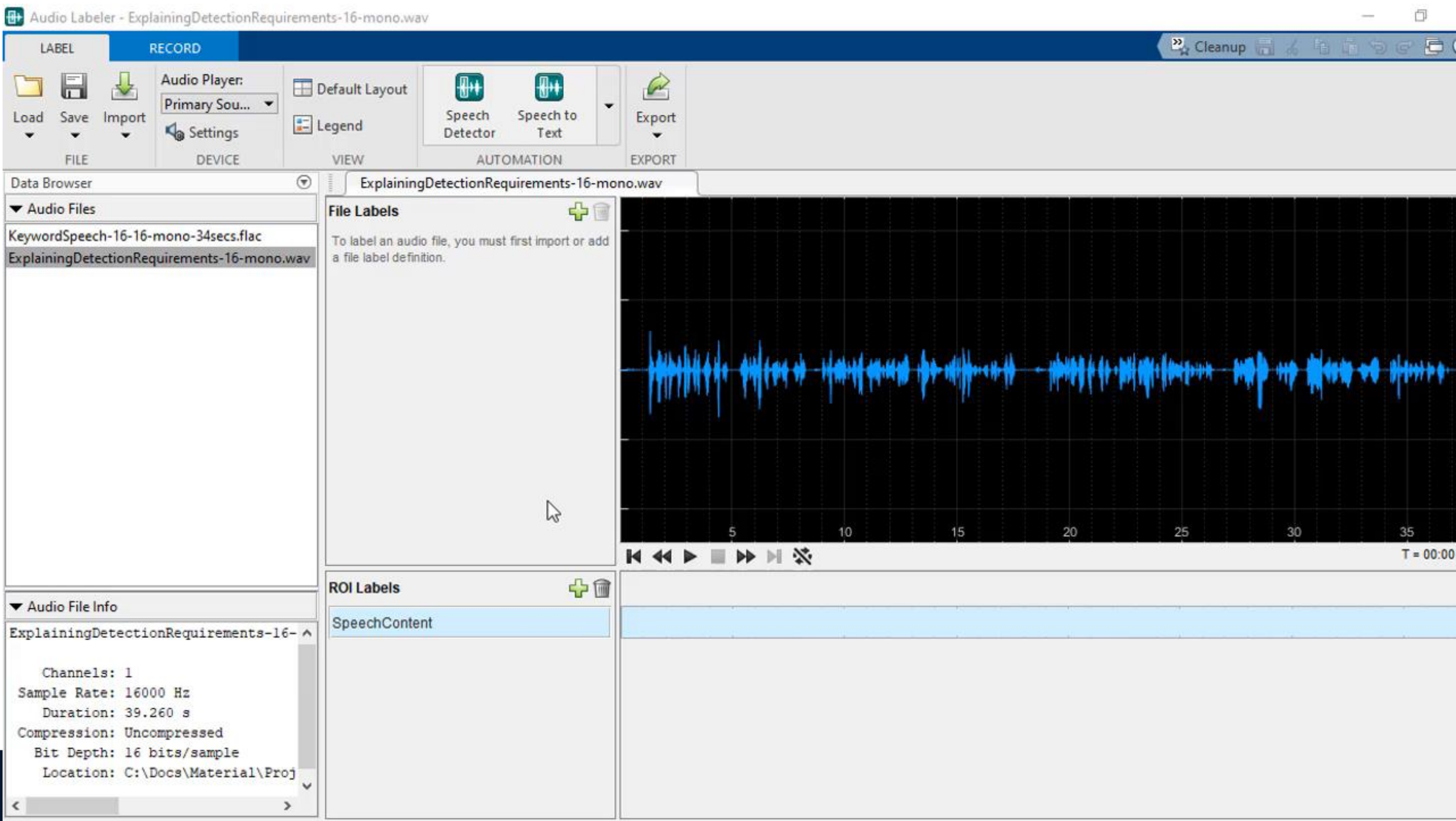

Audio Labeler - ExplainingDetectionRequirements-16-mono.wav
LABEL
RECORD
Cleanup
Load
Save
Import
FILE
Audio Player:
Primary Sou...
Settings
DEVICE
Default Layout
Legend
VIEW
Speech Detector
Speech to Text
AUTOMATION
Export
EXPORT
Data Browser
Audio Files
KeywordSpeech-16-16-mono-34secs.flac
ExplainingDetectionRequirements-16-mono.wav
ExplainingDetectionRequirements-16-mono.wav
File Labels
To label an audio file, you must first import or add a file label definition.
5
10
15
20
25
30
35
T = 00:00:00.000
ROI Labels
SpeechContent
Audio File Info
ExplainingDetectionRequirements-16-
Channels: 1
Sample Rate: 16000 Hz
Duration: 39.260 s
Compression: Uncompressed
Bit Depth: 16 bits/sample
Location: C:\Docs\Material\Proj
Samples Underrun = 0

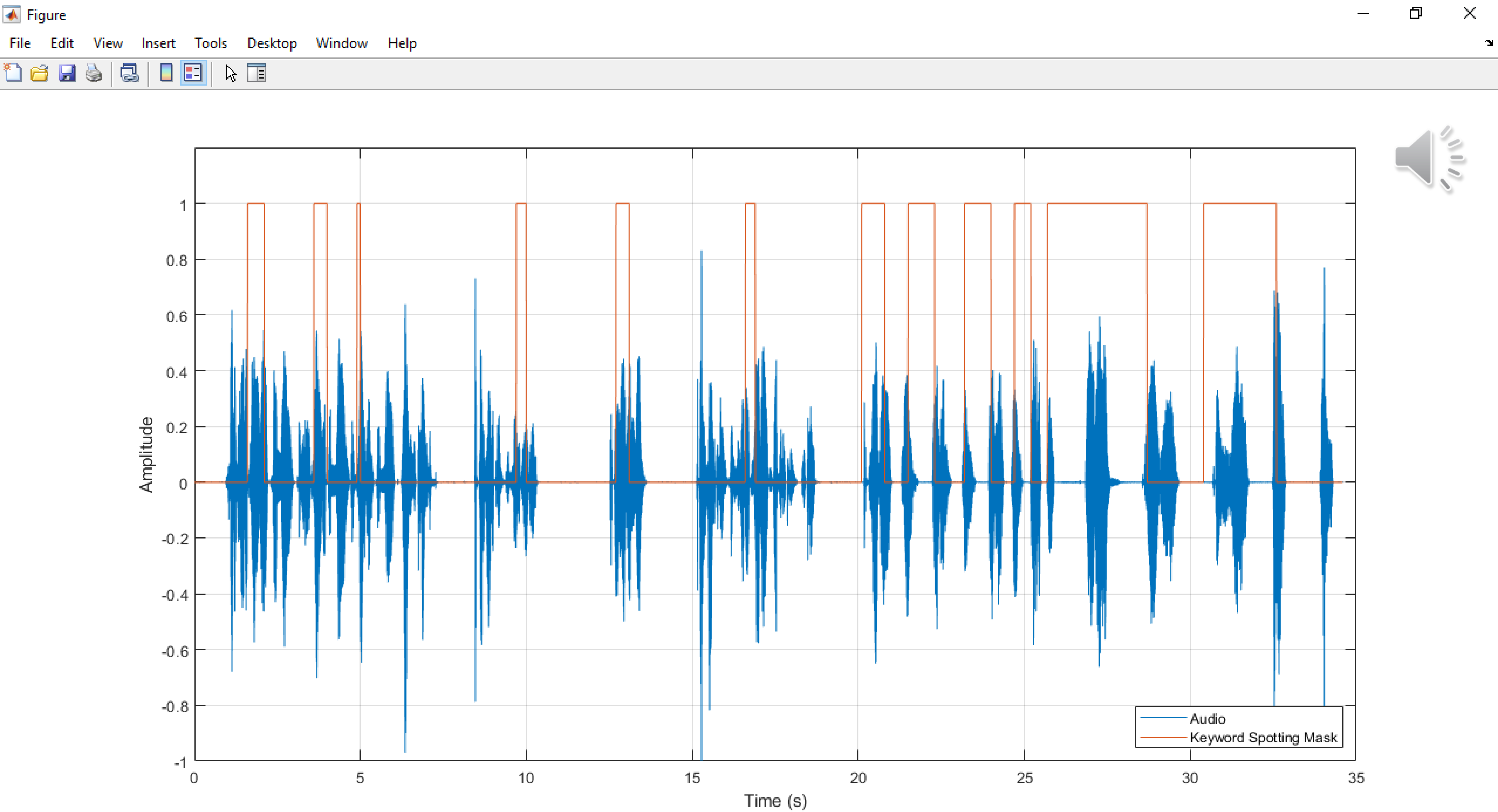
How to label new non-annotated data?

Use an intelligent system trained to carry out a similar tasks with proven accuracy!

For example:

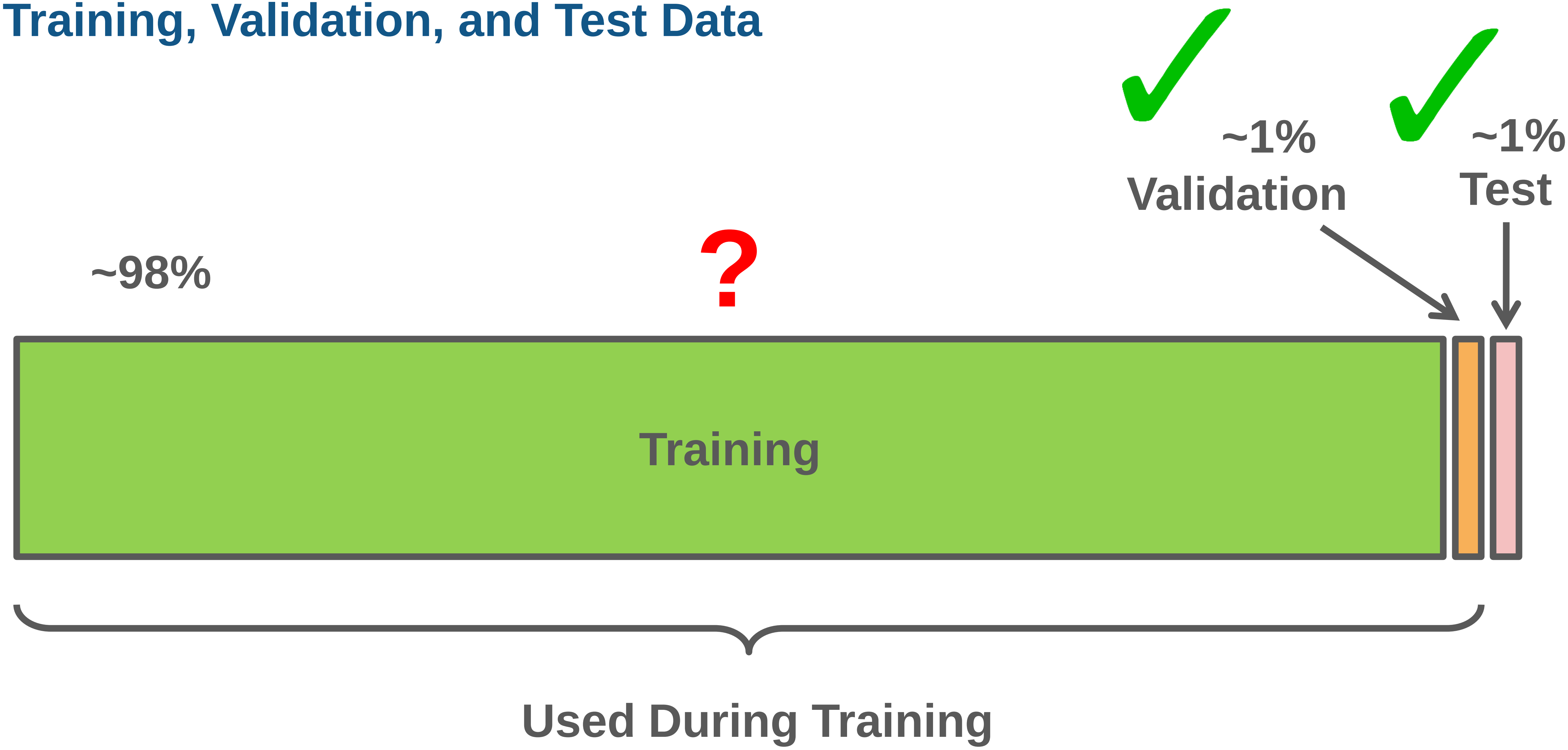
- Humans
- Pre-trained machine learning models

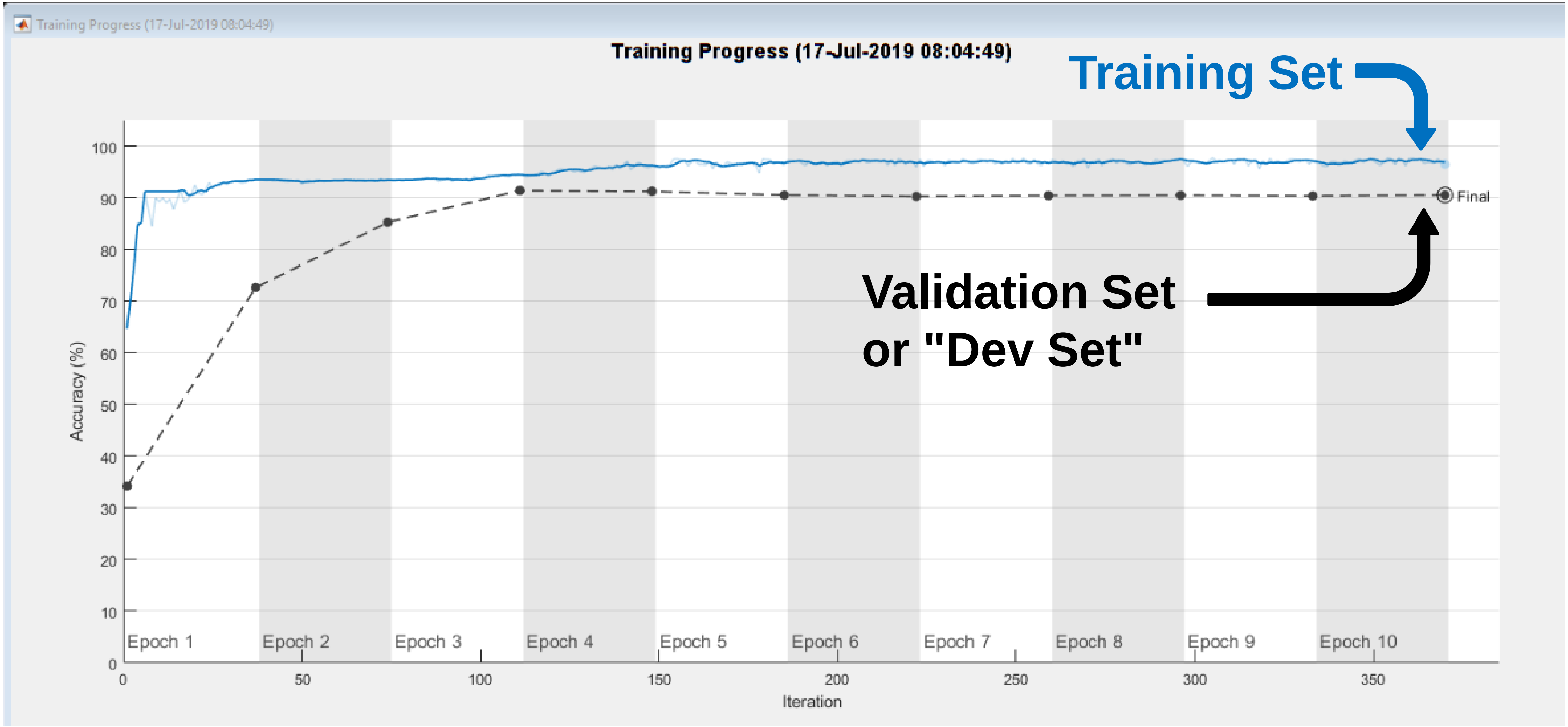

Audio Labeler - ExplainingDetectionRequirements-16-mono.wav
LABEL
RECORD
Cleanup
Load
Save
Import
FILE
Audio Player:
Primary Sou...
Settings
DEVICE
Default Layout
Legend
VIEW
Speech Detector
Speech to Text
AUTOMATION
Export
EXPORT
Data Browser
Audio Files
KeywordSpeech-16-16-mono-34secs.flac
ExplainingDetectionRequirements-16-mono.wav
ExplainingDetectionRequirements-16-mono.wav
File Labels
To label an audio file, you must first import or add a file label definition.
5
10
15
20
25
30
35
T = 00:00
ROI Labels
SpeechContent
Audio File Info
ExplainingDetectionRequirements-16-
Channels: 1
Sample Rate: 16000 Hz
Duration: 39.260 s
Compression: Uncompressed
Bit Depth: 16 bits/sample
Location: C:\Docs\Material\Proj



Full code available here: https://www.mathworks.com/help/audio/ref/audiolabeler-app.html#mw_4e740c85-499f-4087-8d52-95d1b508b7da

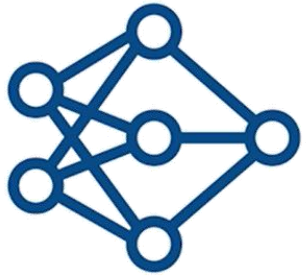
Training, Validation, and Test Data



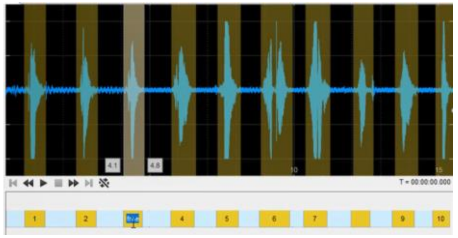


Training Progress (17-Jul-2019 08:04:49)
Training Set
Validation Set
or "Dev Set"
Accuracy (%)
Iteration
Epoch 1
Epoch 2
Epoch 3
Epoch 4
Epoch 5
Epoch 6
Epoch 7
Epoch 8
Epoch 9
Epoch 10
Final

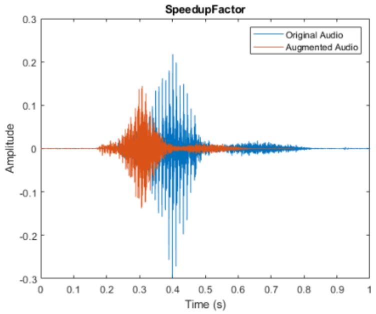
Agenda



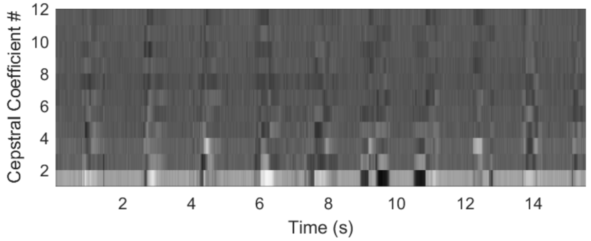
- Basics on training deep neural networks for signals



- Annotating data to train networks for practical applications



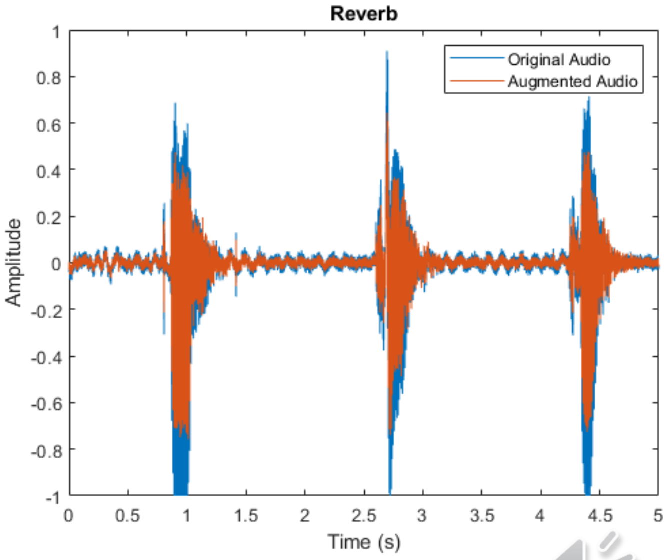
- Generating new data – synthesis and augmentation
- Creating inputs for deep networks



- From system models to real-time prototypes

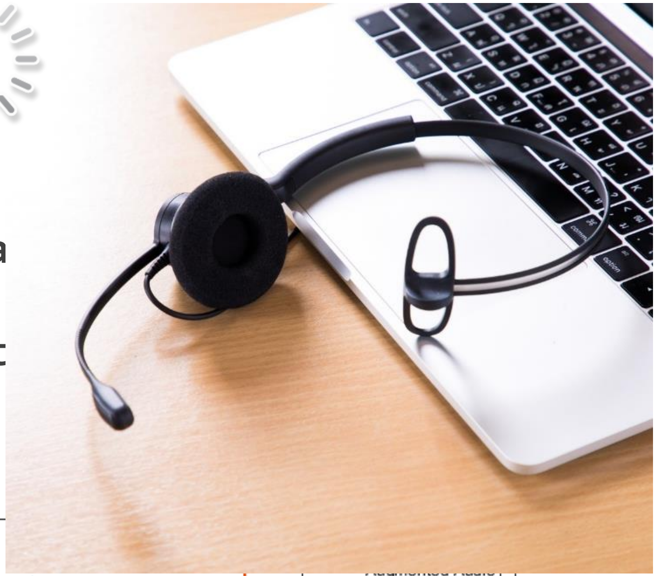
Augmentation – Application-Specific Effects

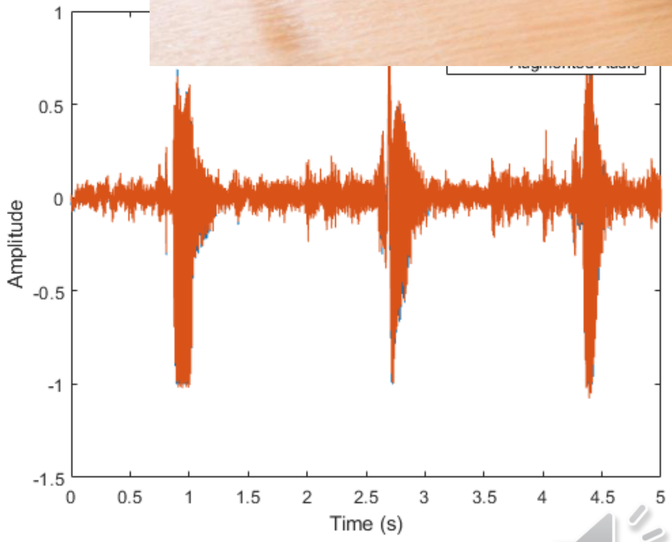
```
>> auAugm.AugmentationInfo
ans =
  struct with fields:

    Reverb: 1
```



Add Kitchen Reverberation

```
>> a
ans
  st
```

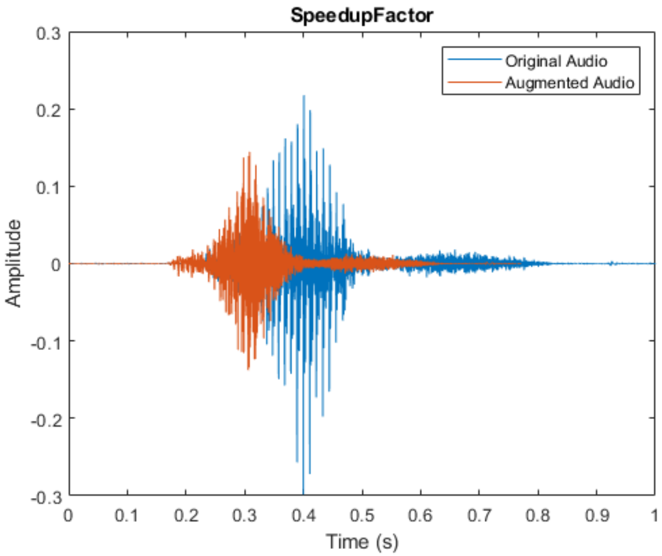




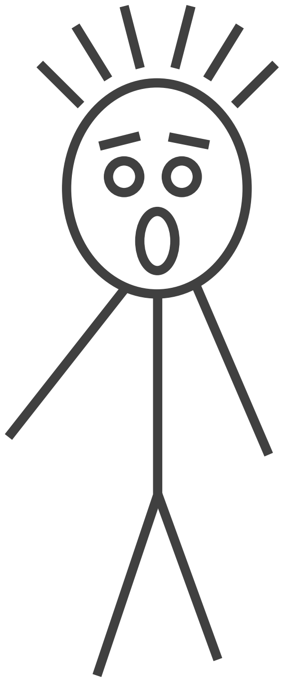
Add Washing Machine Noise

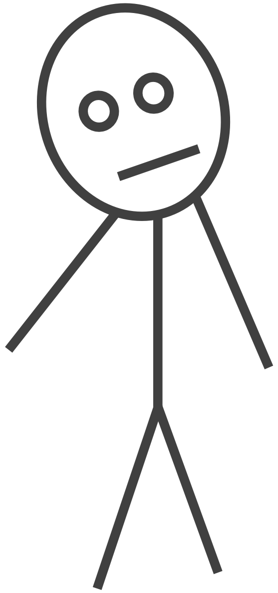
Augmentation – Common Effective Speech Effects

```
>> data.AugmentationInfo(1)

ans = struct with fields:
    SpeedupFactor: 1.3
```

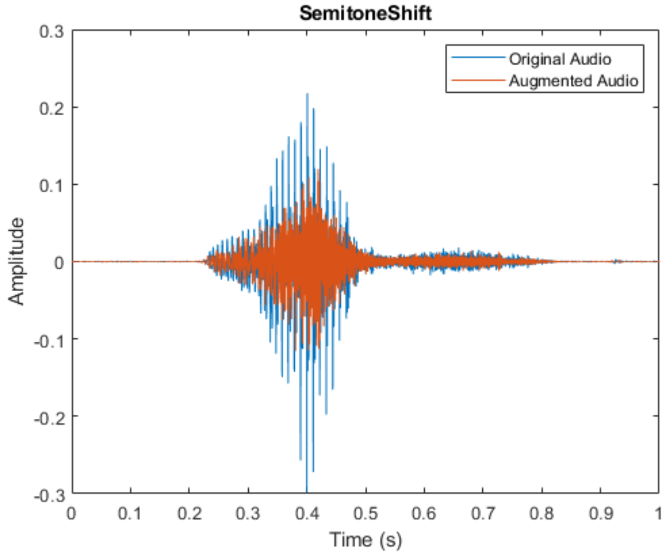


Time Stretching





```
>> data.AugmentationInfo(2)

ans = struct with fields:
    SemitoneShift: -2
```

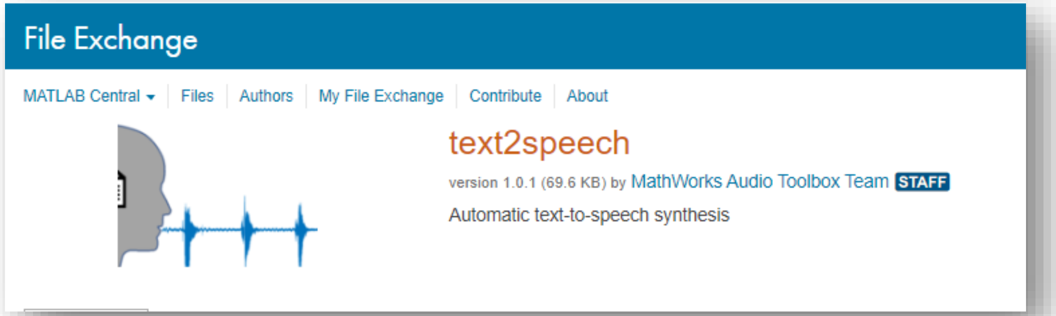


Pitch Shifting

Learn more on [audioDataAugmenter]

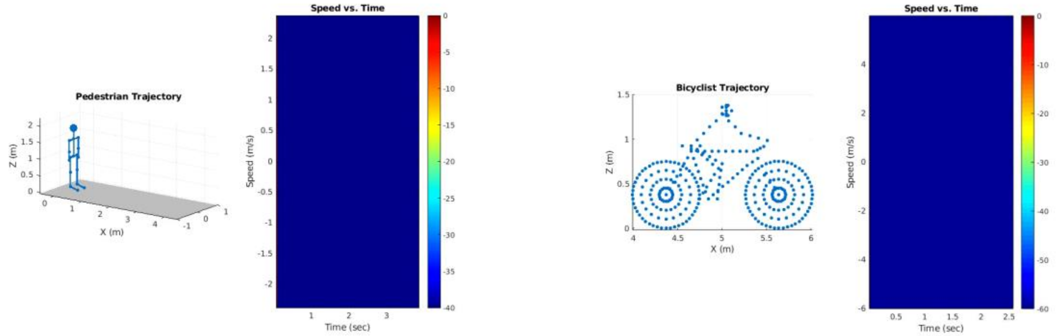
Synthesis – Generative AI models or domain-specific simulations

New `text2speech` function



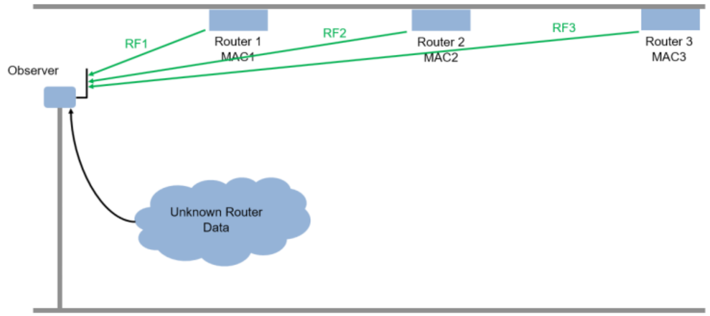
https://www.mathworks.com/matlabcentral/fileexchange/73326-text2speech

Pedestrian and Bicyclist (Radar) Classification



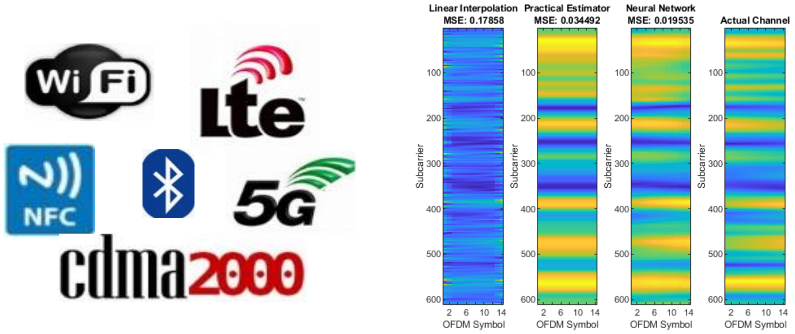
https://www.mathworks.com/help/phased/examples/pedestrian-and-bicyclist-classification-using-deep-learning.html

WLAN Router Impersonation Detection



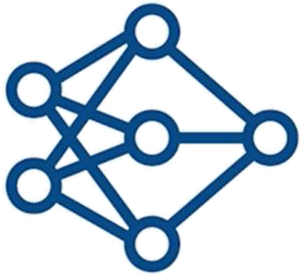
https://www.mathworks.com/help/comm/examples/design-a-deep-neural-network-with-simulated-data-to-detect-wlan-router-impersonation.html

5G Channel Estimation

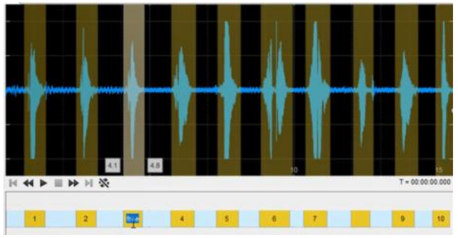


https://www.mathworks.com/help/5g/examples/deep-learning-data-synthesis-for-5g-channel-estimation.html

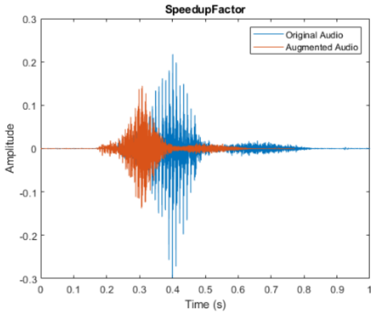
Agenda



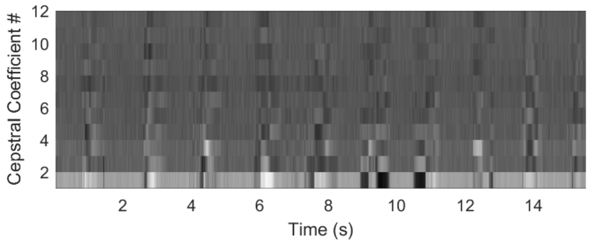
- Basics on training deep neural networks for signals



- Annotating data to train networks for practical applications



- Generating new data – synthesis and augmentation

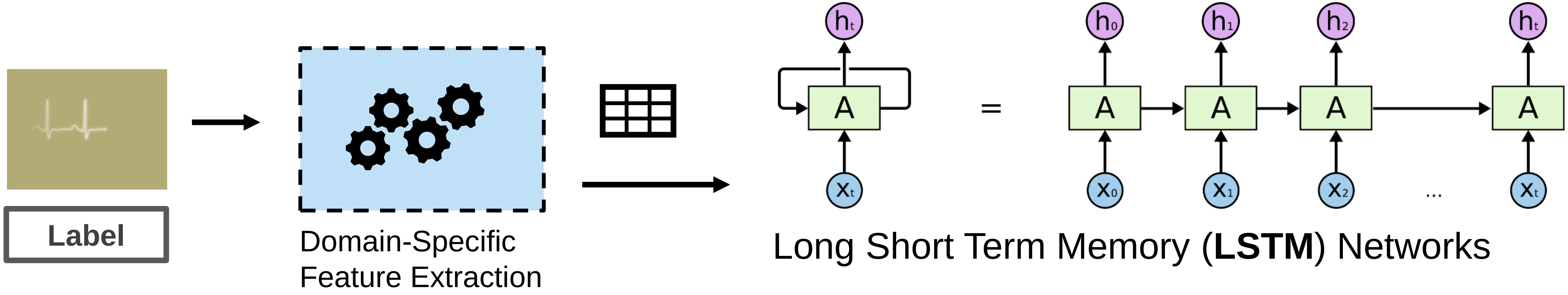


- Creating inputs for deep networks

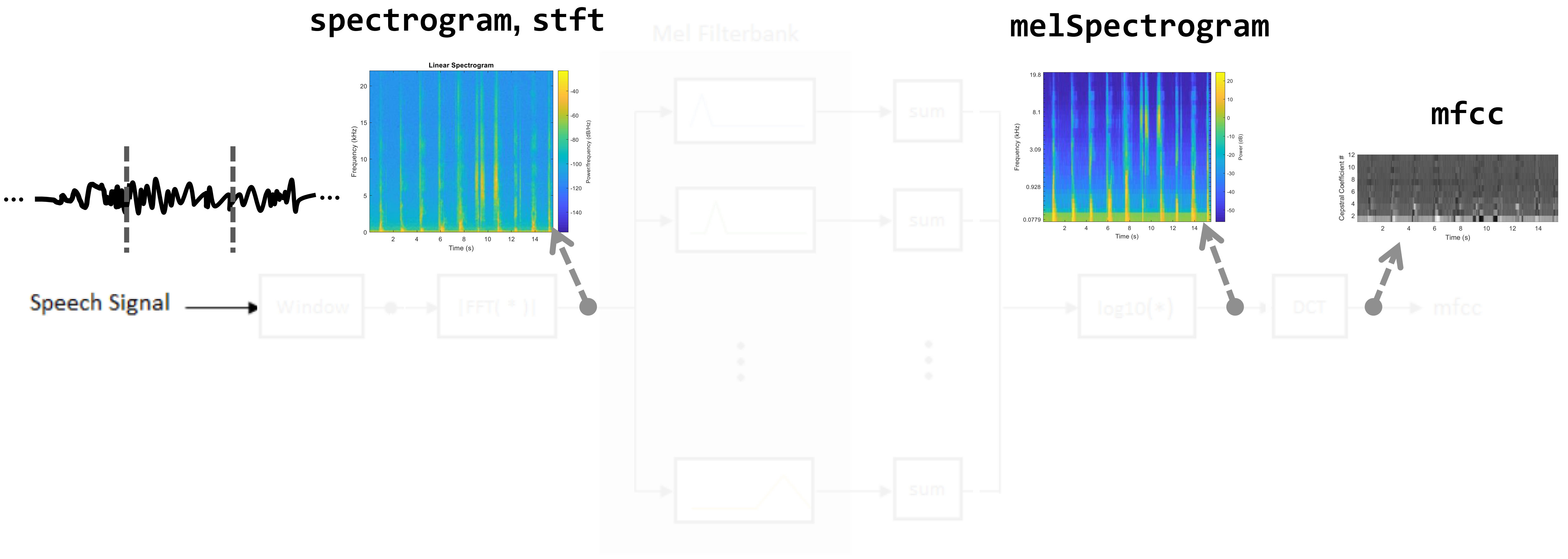
- From system models to real-time prototypes

Training deep networks with time-domain signals most often requires extracting features

Deep learning ≠ End-to-end learning



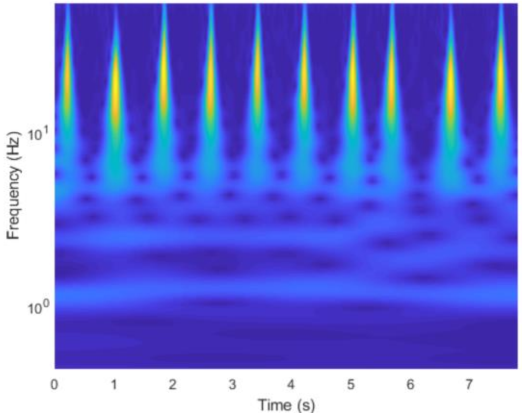
Different applications require different feature extraction techniques



Many other time-frequency transforms and signal features

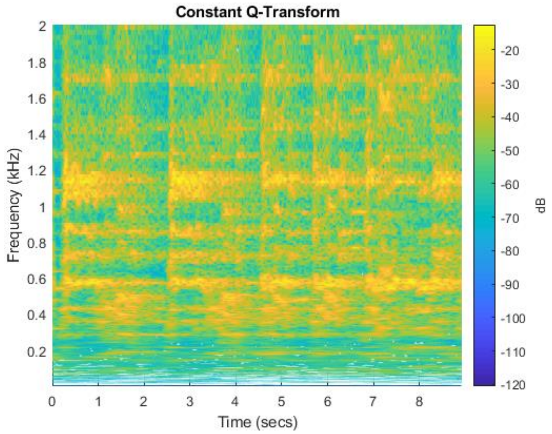
`cwt`

(Continuous wavelet transform)



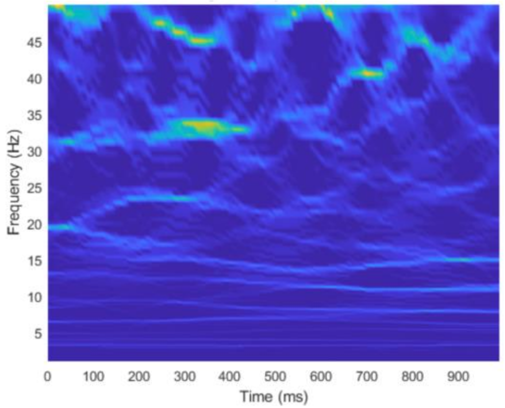
`cqt`

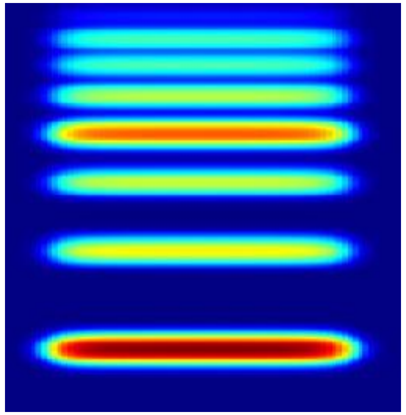
(Constant Q transform)



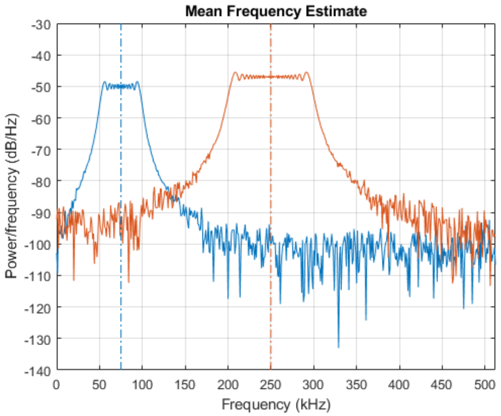
`wsstridge`

(Synchrosqueezing)

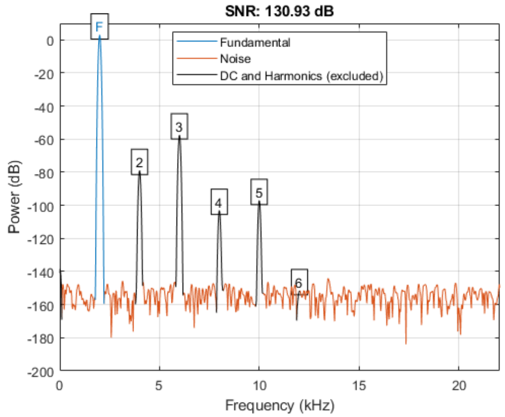




`waveletScattering`

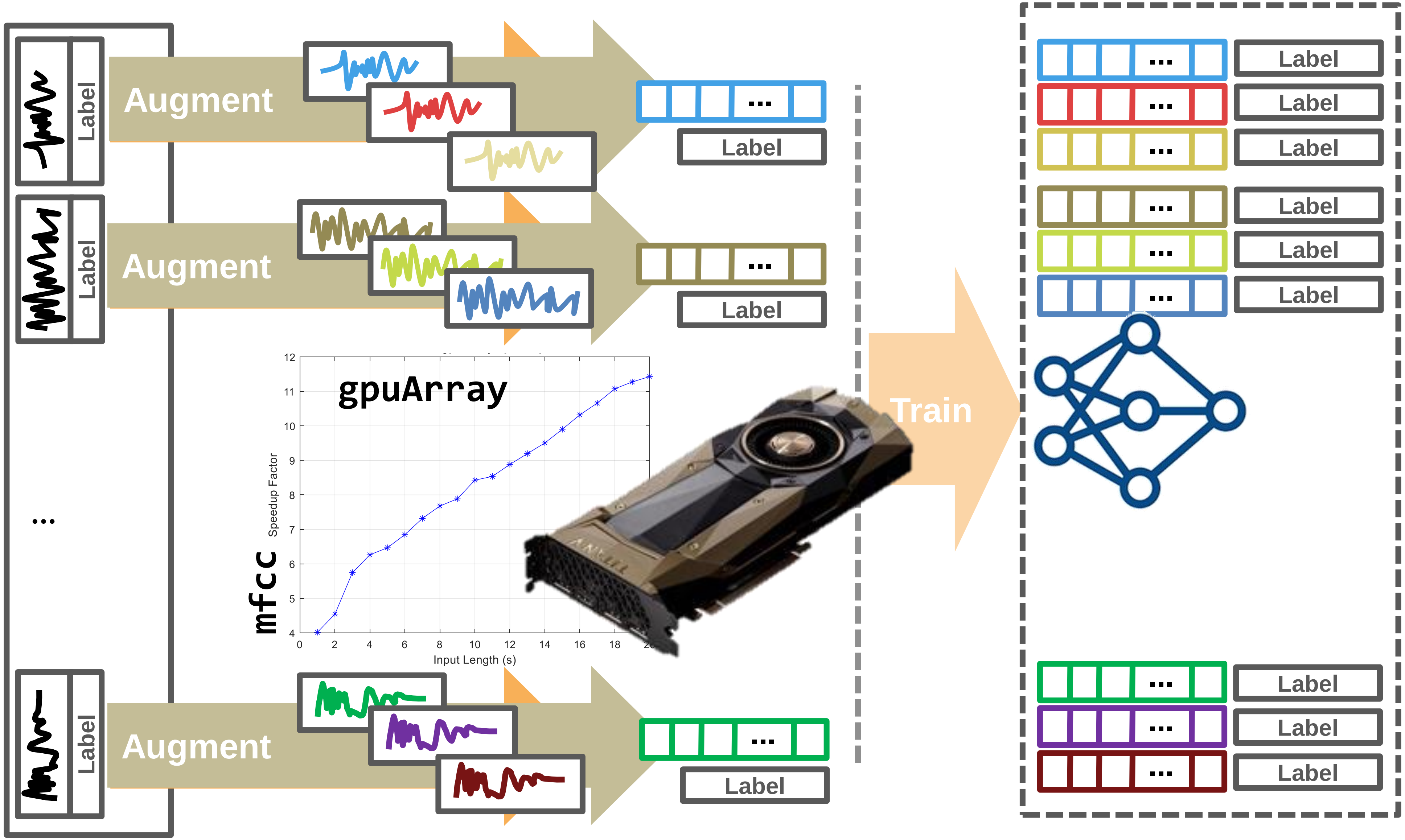


(Spectral statistics)

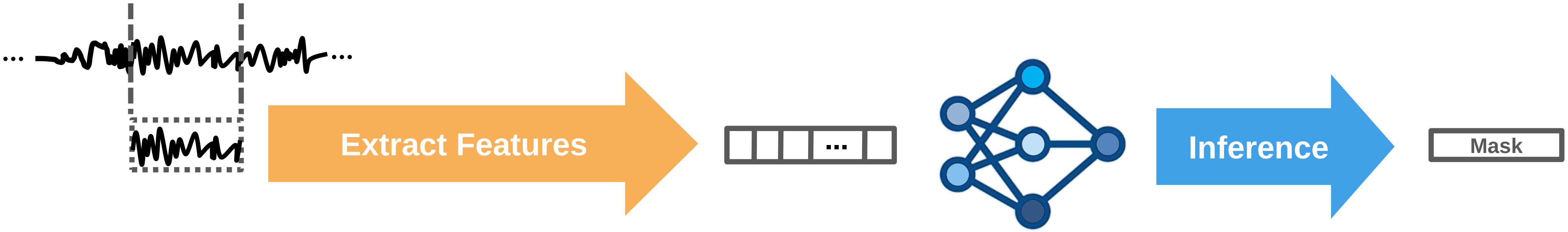


(Harmonic analysis)

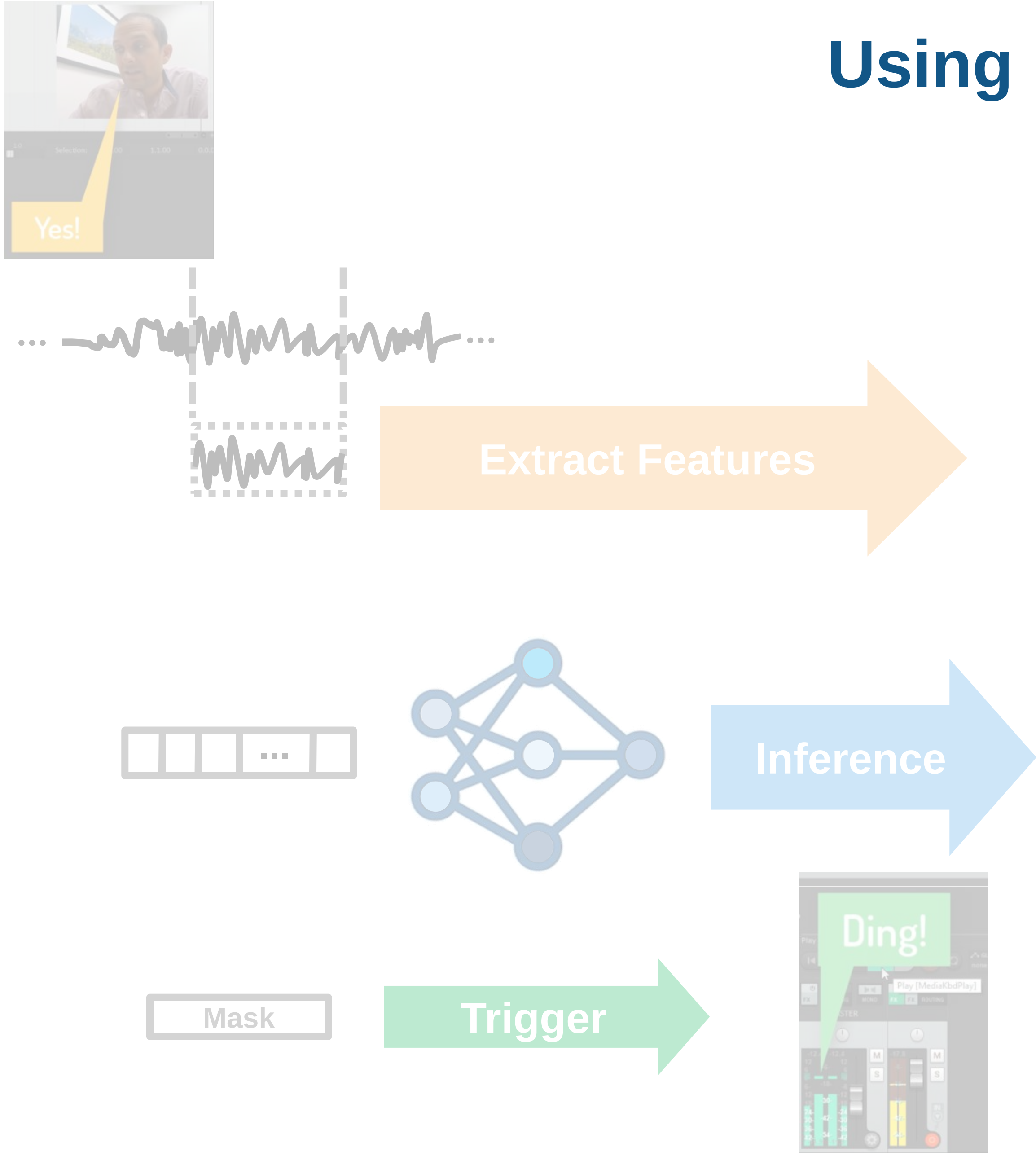
Providing Input Data for Network Training



Using Network for Prediction (aka Inference)



Using Network for Prediction (aka Inference)

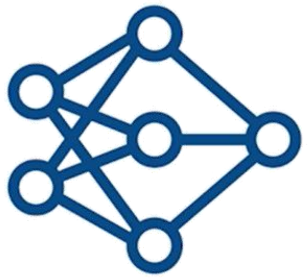



```
[...]
% Exctract MFCC from whole analysis buffer
[coeffs,delta,deltaDelta] = mfcc(buf,SampleRate,...
    'WindowLength',winLength,...
    'OverlapLength',ovlpLength);

% Concatenate and normalize features
featureMatrix = [coeffs,delta,deltaDelta];
featureMatrix = (featureMatrix - M)./S;
```

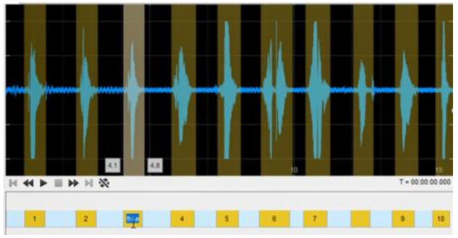
```
% Detect keyword with LSTM network (Mask around speech keyword)
featMask    = classify(net,featureMatrix.');
```

```
% Debounce and re-align detections in time domain
[timeMask, chimePosition] = debounceAnalyzeDetectionMask(featMask);

% Generate chimes for detection events
chime = generateChimeAtSample(chimePosition,...
[...]
```

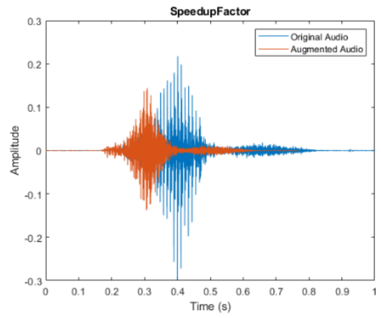
Agenda



- Basics on training deep neural networks for signals

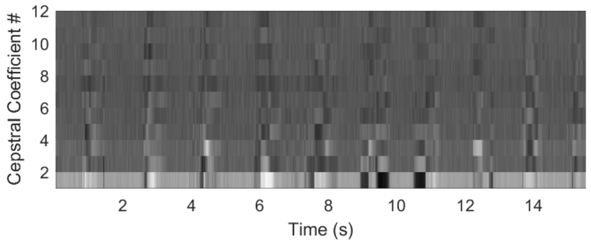


- Annotating data to train networks for practical applications

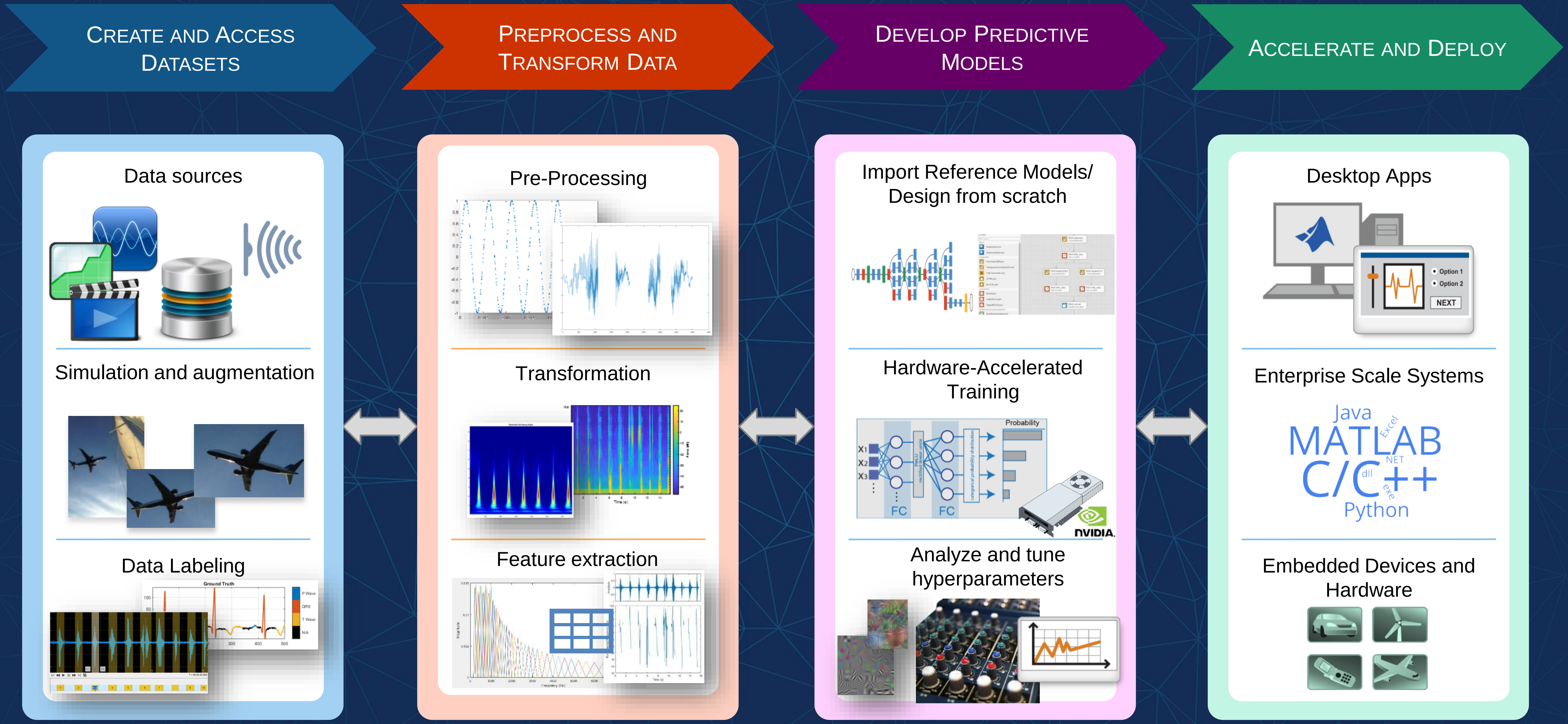


- Generating new data – synthesis and augmentation

- Creating inputs for deep networks

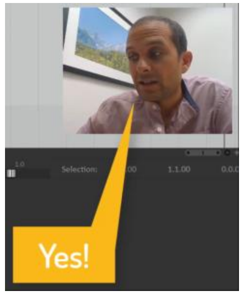


- From system models to real-time prototypes

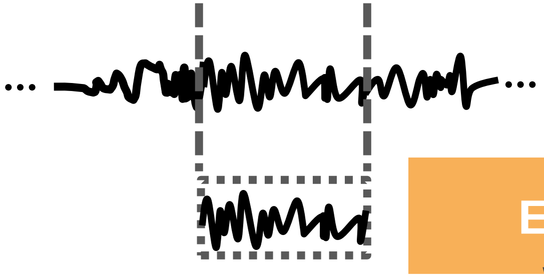


Create and Access Datasets
Preprocess and Transform Data
Develop Predictive Models
Accelerate and Deploy
Data sources
Simulation and augmentation
Data Labeling
Ground Truth
Pre-Processing
Transformation
Feature extraction
Import Reference Models/ Design from scratch
Hardware-Accelerated Training
Probability
FC
FC
NVIDIA.
Analyze and tune hyperparameters
Desktop Apps
Option 1
Option 2
NEXT
Enterprise Scale Systems
Java
Excel
MATLAB
.NET
C/C++
dll
exe
Python
Embedded Devices and Hardware

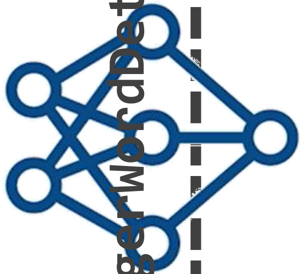
Using Network for Prediction (aka Inference)




```
>> generateAudioPlugin triggerWordDetector
```



Extract Features



Inference

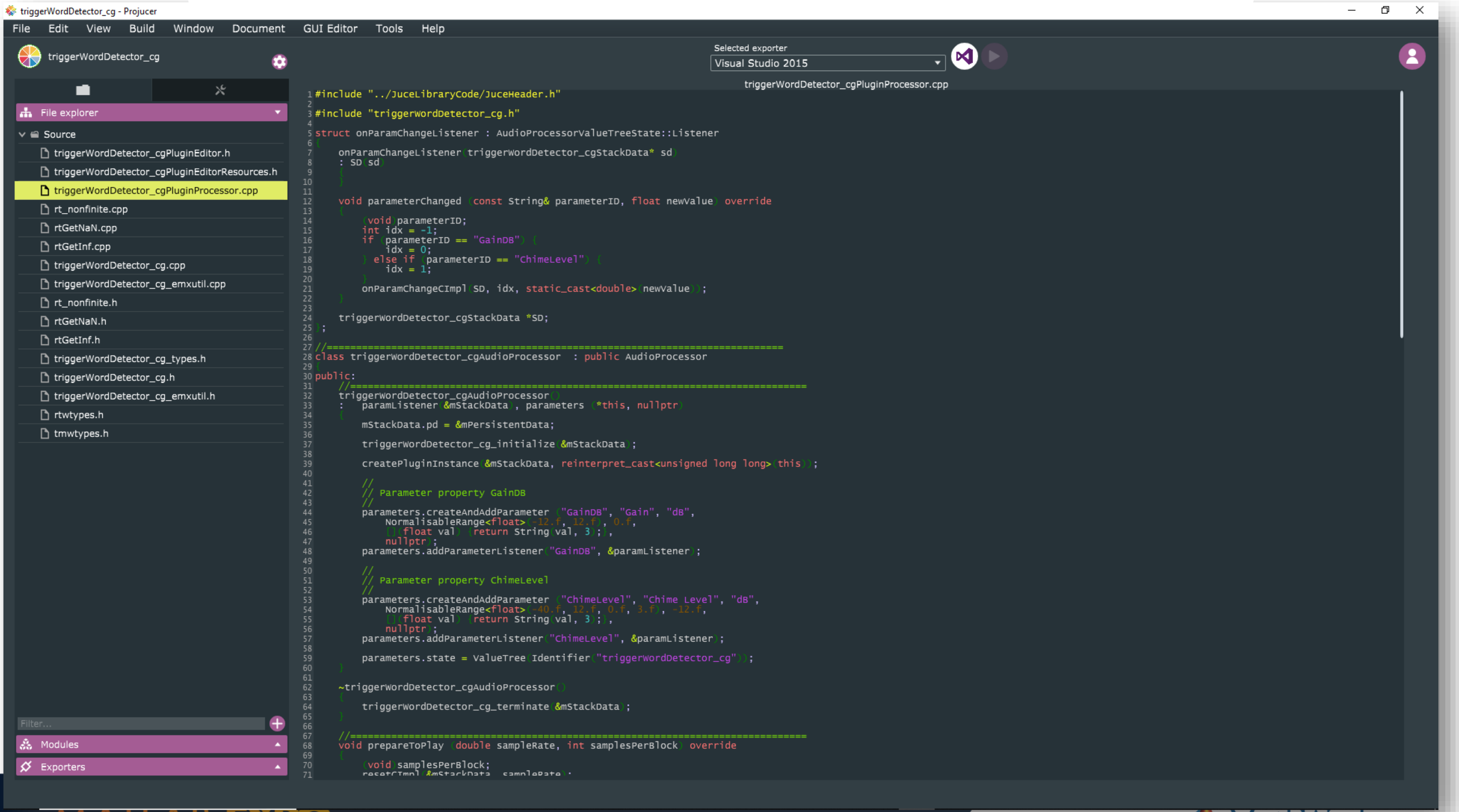
Mask

Trigger

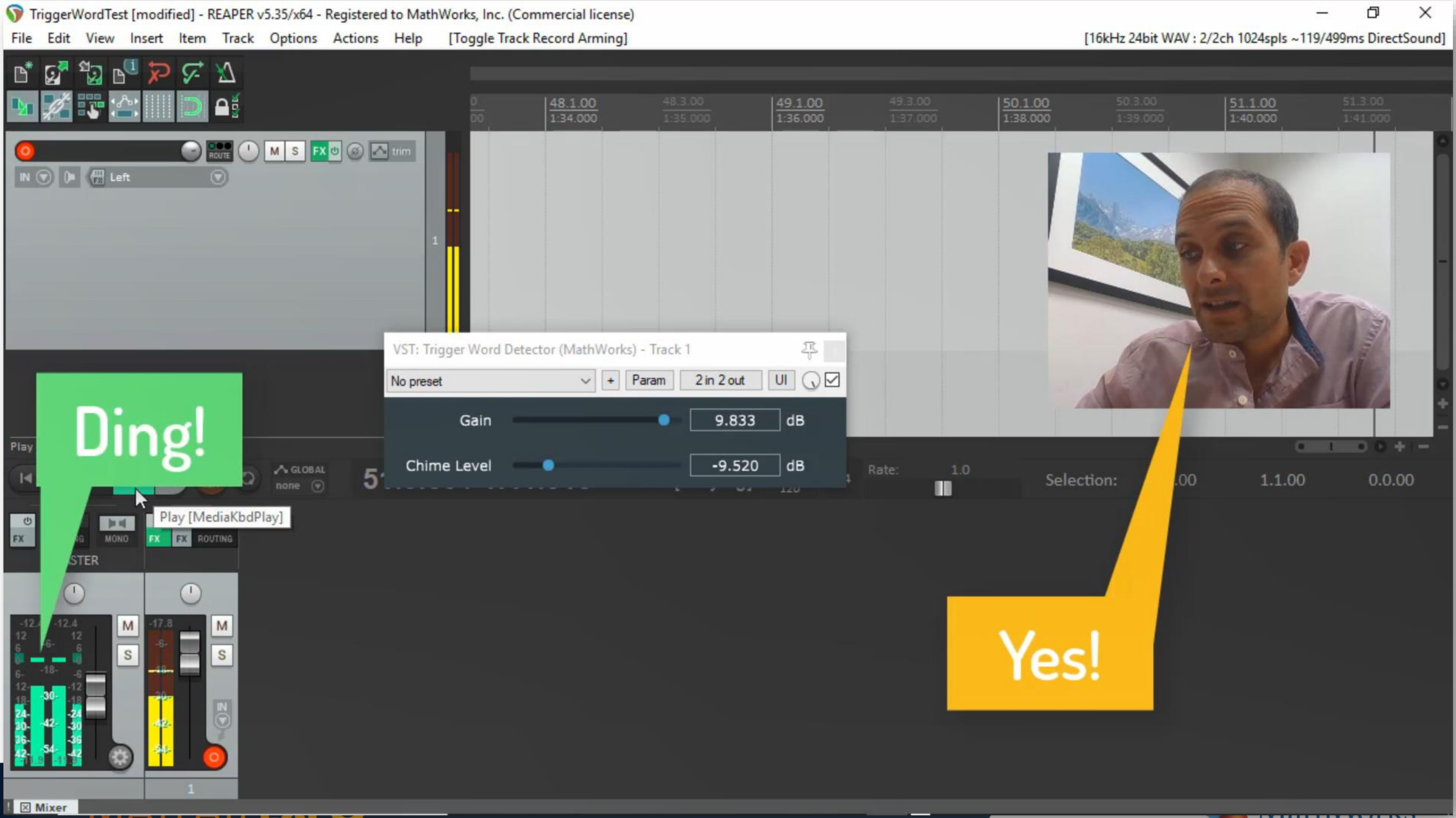
triggerWordDetector.m

```
[...]

% Exctract MFCC from whole analysis buffer
[coeffs,delta,deltaDelta] = mfcc(buf,SampleRate,...
    'WindowLength',winLength,...
    'OverlapLength',ovlpLength);

% Concatenate and normalize features
featureMatrix = [coeffs,delta,deltaDelta];
featureMatrix = (featureMatrix - M)./S;

% Detect keyword with LSTM network (Mask around speech keyword)
featMask    = classify(net,featureMatrix.');

% Debounce and re-align detections in time domain
[timeMask, chimePosition] = debounceAnalyzeDetectionMask(featMask)

% Generate chimes for detection events
chime = generateChimeAtSample(chimePosition,...
[...]
```

\>\> generateAudioPlugin triggerWordDetector

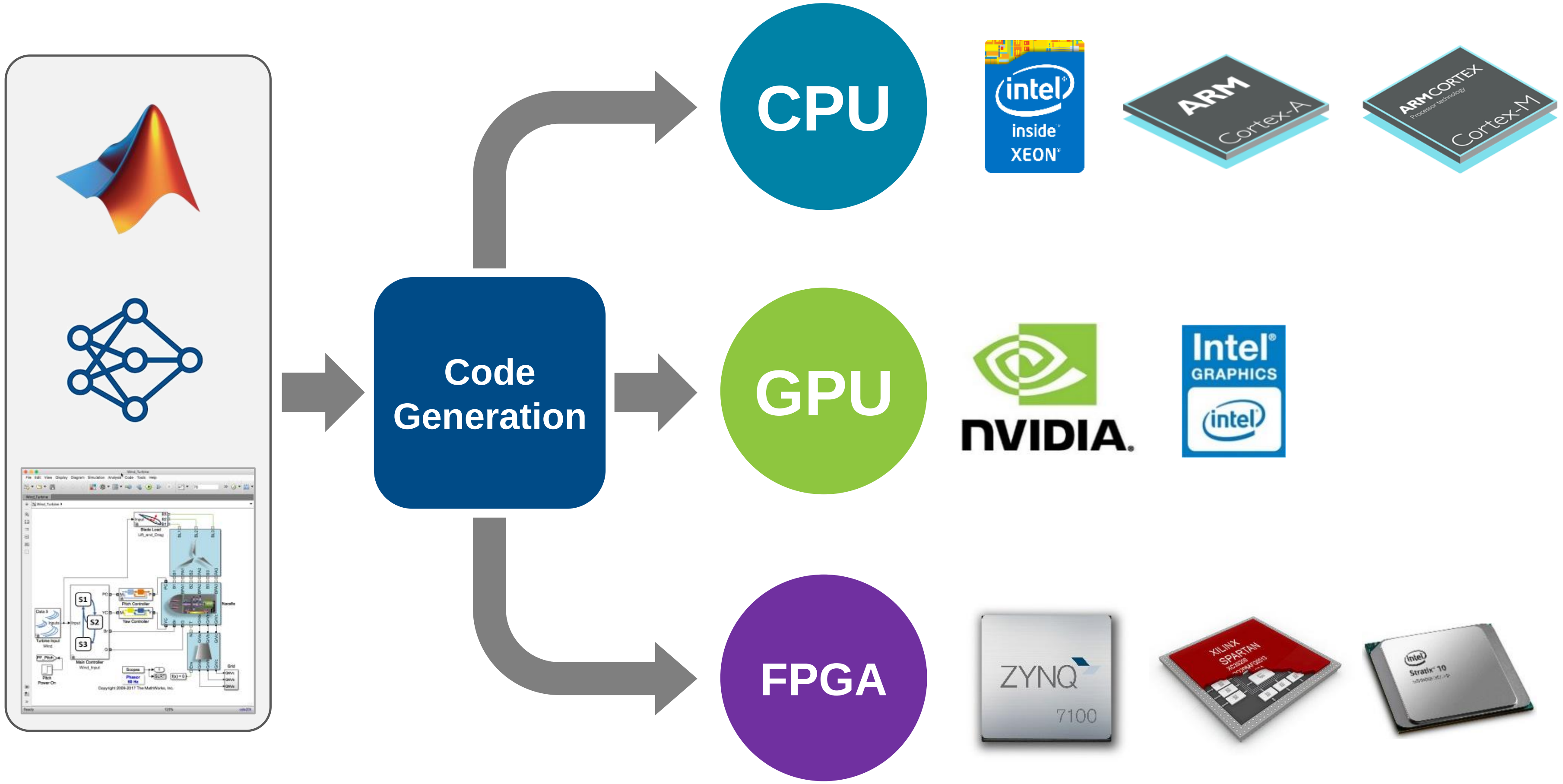


```
>> generateAudioPlugin triggerWordDetector
```



Deploy to any processor with best-in-class performance

AI models in MATLAB and Simulink can be deployed on embedded devices, edge devices, enterprise systems, the cloud, or the desktop.



Q: "What do I need to develop such a system?"

A: "A simple and proven deep learning model"

A: "A lot of data, a good dose of signal processing expertise, and the right tools for the specific application in hand"

Deep learning systems can only be as good as the data used to train them

mathworks.com