Q&A

Please fill the session-related questions into the Q&A sheet in your registration kit and hand over the sheet to our promoters
Developing and Deploying Machine Learning Solutions for Embedded Applications

Nitin Rai – Application Engineer
Machine Learning Solutions for Embedded Applications

Application examples

- Fitness Trackers
- Structural health monitoring (SHM)
- Fault and event detection
- Advanced surveillance
- Medical Devices
- Face detection
- .......
Machine Learning Solutions for Embedded Applications

Challenges

- Data access from multiple sensors on embedded devices
- Iterative feature extraction and model development
- Tuning model for embedded deployment
- Implementing and deploying models on embedded devices
Challenge
Develop automated software for detecting oversteering, an unsafe condition in which rear tires lose their grip during a turn.

Solution
Use MATLAB to develop, train, and evaluate a variety of supervised machine learning classifier types, including KNN, SVM, and decision trees.

Results
- Oversteering identified with greater than 98% accuracy
- Multiple machine learning classifiers trained automatically
- Code generated and deployed to an ECU for real-time, in-vehicle testing

“Working in MATLAB, we developed a supervised machine learning model as a proof of concept. Despite having little previous experience with machine learning, in just three weeks we completed a working ECU prototype capable of detecting oversteering with over 98% accuracy.”
- Tobias Freudling, BMW Group
Online Health Monitoring Using Vibration Data

<table>
<thead>
<tr>
<th>Platform</th>
<th>Raspberry Pi</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data</td>
<td>Vibration Data</td>
</tr>
<tr>
<td>Sensor</td>
<td>Sense HAT: Accelerometer along x, y, and z axes</td>
</tr>
<tr>
<td>Prediction</td>
<td>Stopped</td>
</tr>
<tr>
<td></td>
<td>Normal</td>
</tr>
<tr>
<td></td>
<td>Blocked</td>
</tr>
<tr>
<td></td>
<td>Imbalanced</td>
</tr>
<tr>
<td>Output</td>
<td>On ThingSpeak</td>
</tr>
</tbody>
</table>

![Diagram of vibration data monitoring and prediction with Raspberry Pi and Sense HAT](image)

1. Stop Mode

![Screenshot of ThingSpeak interface with vibration data charts](image)
Developing and Deploying Machine Learning Workflow

1. Sensor Data Acquisition
2. Feature Extraction
3. Develop Predictive Model
4. Embedded Implementation
5. Cloud based Communication

- RASPBERRYPI
- Sensor data acquisition
- Feature extraction
- Predictive model development
- Embedded implementation
- Cloud-based communication
Access and Explore Sensor Data

- Raspberry Pi support package lets you Acquire sensor and image data from your connected Raspberry Pi into MATLAB and SIMULINK

https://www.mathworks.com/hardware-support/raspberry-pi-matlab.html

mysh = sensehat(mypi)

acceleration = readAcceleration(mysh)
Access and Explore Sensor Data

- MATLAB Support Packages for Apple iOS Sensors and Android Sensors
- MATLAB Support Package for Arduino Hardware
- Data Acquisition Toolbox - Connect to data acquisition cards, devices, and modules
Developing and Deploying Machine Learning Workflow

1. **Sensor Data Acquisition**
   - Raspberry Pi
   - Accel

2. **Feature Extraction**
   - accel_x, accel_y, accel_z
   - feature1, feature2

3. **Develop Predictive Model**
   - fitdata, label
   - classify_op_mode

4. **Embedded Implementation**

5. **Cloud based Communication**
   - RASPBERRYPI
   - user1, user2, user3, user4
   - Health Gauge
   - Abnormal
   - Ok
   - BLOCK

- MathWorks
- ThingSpeak
function [featmat, feature1, feature2, feature3] = extractfeat(accx_mat, accy_mat, accz_mat)

%% Feature Extraction

featmat = zeros(1,33);
fs = 10.0;

%% Mean
feature1 = mean(accx_mat, 1);
%feature1 = feature1();
featmat(1) = feature1;

%%
feature2 = mean(accy_mat, 1);
%feature2 = feature2();
featmat(2) = feature2;

%%
feature3 = mean(accz_mat, 1);
%feature3 = feature3();
featmat(3) = feature3;

%% Root Mean Square
feature4 = rms(accx_mat, 1);
%feature4 = feature4();
featmat(4) = feature4;

%%
Domain-Specific Features and Transformations – Examples

**Speech and Audio**
- MFCC
- GTCC
- MDCT
- Pitch, harmonicity
- Spectral shape descriptors
- ...

**Navigation and Sensor Fusion**
- Orientation
- Height
- Position
- Multi-object tracking
- ... from
- Acceleration, angular velocity
- Magnetic field
- GPS reading

**Radar**
- Micro-Doppler analysis
- Range-Doppler processing
- Synthetic aperture imaging
- Spectral analysis
- Waveform ambiguity
- ...

**Text Analytics**
- Train Word Embeddings
- Word2Vec
- Topic Modeling
- ...

**Acoustics**
- Microphone sound
- Speaker identity
- Room acoustics
- Audio processing
- ...

**Imaging**
- Image segmentation
- Object detection
- Video analysis
- Image recognition
- ...

**GPS**
- Location
- Navigation
- Tracking
- ...
Developing and Deploying Machine Learning Workflow

Sensor Data Acquisition → Feature Extraction → Develop Predictive Model → Embedded Implementation → Cloud based Communication
>> classificationLearner
Perform feature selection using Neighborhood Component Analysis

\texttt{fscnca(X,Y)} performs feature selection for classification using the predictors in \( X \) and responses in \( Y \).

**Principal component analysis**

- To emphasize variation and bring out strong patterns in a dataset

**Bayesian Optimization**

- Tune hyperparameters of machine learning algorithms automatically
Developing and Deploying Machine Learning Workflow

1. Sensor Data Acquisition
   - RASPBERRYPI
   - Accel

2. Feature Extraction
   - accel_x, accel_y, accel_z
   - extraint features

3. Develop Predictive Model
   - predict data
   - classify op mode

4. Embedded Implementation

5. Cloud based Communication
   - RASPBERRYPI
   - user1
   - user2
   - user3
   - user4

- Health Gauge
  - Abnormal
  - OK
  - BLOCK

- ThingSpeak
Embedded Implementation

DSP / MCU

Embedded Coder

C, C++

MATLAB®

SIMULINK®
Why Automatic Code Generation?

Challenges with manual coding:

- Separate functional and implementation specification
  - Leads to multiple implementations that are inconsistent
  - Hard to modify requirements during development
- Manual coding errors
- Time-consuming and expensive process

With automatic code generation, design engineers can:

- Maintain one design across simulation and implementation
- Design faster and get to C
- Test more systematically and frequently
- Spend more time improving algorithms
Fault Detection Using Sense HAT
Fault Detection Using Sense HAT
Code Generation Products for C/C++

**MATLAB® Coder**
Automatically generate C and C++ from MATLAB code

**Simulink® Coder™**
Automatically generate C and C++ from Simulink models and Stateflow charts

**Embedded Coder™**
Automatically generate C and C++ optimized for embedded systems
Connecting MATLAB and Simulink to Hardware

- Android and iOS
- Arduino® Uno, Mega 2560
- LEGO® MINDSTORMS® NXT
- Raspberry Pi Model B
- BeagleBoard-xM
- PandaBoard
- BeagleBone Black
- RTL-SDR

Hardware Support
Simulink Support Package for Android Devices

- Interactive parameter tuning and signal monitoring
- Model deployment for standalone operation
- Simple UI using sliders and buttons
- Generation of Android Studio compatible projects
Simulink Support Package for Raspberry Pi

- Interactive parameter tuning and signal monitoring
- Model deployment for standalone operation
Key Takeaways

✔ Data access from multiple sensors on embedded device
✔ Iterative feature extraction and Machine learning model development
✔ Tuning model for embedded deployment
✔ Implementing and deploying models on embedded devices
This two-day course focuses on data analytics and machine learning techniques in MATLAB. The course demonstrates the use of unsupervised learning to discover features in large data sets and supervised learning to build predictive models. Topics include:

- Organizing and preprocessing data
- Clustering data
- Creating classification and regression models
- Interpreting and evaluating models
- Simplifying data sets
- Using ensembles to improve model performance
Embedded Coder for Production Code Generation

This hands-on, three-day course focuses on developing models in the Simulink® environment to deploy on embedded systems. The course is designed for Simulink users who intend to generate, validate, and deploy embedded code using Embedded Coder®.

Topics include:

- Generated code structure and execution
- Code generation options and optimizations
- Integrating generated code with external code
- Generating code for multirate systems
- Customizing generated code
- Customizing data
- Deploying code
MATLAB EXPO 2019

Email: nitinrai@mathworks.com
LinkedIn: https://www.linkedin.com/in/nitinrai0111/