MATLAB을 활용한 임베디드 및 프로덕션 시스템으로의 AI 배포
장규환 차장
Using the concept of convective heat transfer to estimate the amount of oil in a transformer tank
Deployment to Embedded and Enterprise Systems

**Enterprise**

Health Monitoring of Distribution Transformers

*SIEMENS*

**Embedded**

Card to Classify Blood Type

*IDNEO*
Agenda

Deploying AI to production is difficult

Three specific challenges:
1. Limitations of Embedded hardware
2. Ongoing changes in environment or system behavior
3. Scale to production load in Enterprise systems
Two Approaches for integrating AI with Larger System

MATLAB

Code Generation

Embedded Systems

CPU
GPU
FPGA

Compiler

Enterprise Systems
Embedded Deployment of Acoustic Scene Recognition

- Squeezenet ~5MB
- ResNet-50 ~100MB

Limited resources
Quiz: Which Sounds do you hear?
Embedded Deployment of Acoustic Scene Recognition

Reformat the data

Convolutional Neural Networks (CNN)

SqueezeNet ~5MB
ResNet-50 ~100MB

Limited resources
How can Embedded Deployment Be Enabled?

Original

Layer Fusion

Pruning

Quantizing
Deep Learning Quantization: Acoustic Scene Classification

Use Deep Network Quantizer to Optimize the Inference Network

```matlab
load('trainedNet');
analyzeNetwork(trainedNet);
numData = size(xTrain);
numData = numData(end);
augImds = augmentedImageDatastore(trainedNet.Layers(1).InputSize, xTrain, yTrain);
calDS = augImds.subset(1:floor(numData * 0.8));
valDS = augImds.subset(floor(numData * 0.8)+1:numData);
dq = dLquantizer(trainedNet, 'ExecutionEnvironment', 'GPU');
dq.calibrate(calDS)
```

- Load trained network
- Split data: calibration – 80%, validation – 20%
- Launch Deep Network Quantizer App
Deep Learning Quantization: Acoustic Scene Classification
Deep Learning Quantization: Acoustic Scene Classification
Deep Learning Quantization: Acoustic Scene Classification

Validation Results

Memory (MB)

<table>
<thead>
<tr>
<th></th>
<th>FP32</th>
<th>INT-8</th>
</tr>
</thead>
<tbody>
<tr>
<td>Learnable Parameters</td>
<td>4.7</td>
<td>1.2</td>
</tr>
<tr>
<td>Top-2 Accuracy</td>
<td>90%</td>
<td>89%</td>
</tr>
</tbody>
</table>

Top-2 Accuracy

<table>
<thead>
<tr>
<th></th>
<th>FP32</th>
<th>INT-8</th>
</tr>
</thead>
<tbody>
<tr>
<td>Learnable Parameters</td>
<td>75%</td>
<td></td>
</tr>
<tr>
<td>Top-2 Accuracy</td>
<td>0.3%</td>
<td></td>
</tr>
</tbody>
</table>
Deep Learning Quantization: Acoustic Scene Classification

Validation Results

<table>
<thead>
<tr>
<th>Learnable Parameters</th>
<th>Memory (MB)</th>
<th>Top-2 Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>FP32</td>
<td>INT-8</td>
<td></td>
</tr>
<tr>
<td>4.7</td>
<td>1.2</td>
<td></td>
</tr>
<tr>
<td>75%</td>
<td>90%</td>
<td>89%</td>
</tr>
</tbody>
</table>

Top-2 Accuracy

- FP32: 90%
- INT-8: 89%

Learnable Parameters

- FP32
- INT-8

Code Generation

- CPU
- GPU
- FPGA
Deploying AI to Embedded and Enterprise systems is difficult

Three specific challenges:
1. Limitations of Embedded hardware
2. Ongoing changes environment or system behavior
3. Scale to production load in Enterprise systems
AI models reflect System behaviors and Environment

(illustration only; not based on actual data)
AI models reflect System behaviors and Environment
Deployed Models Need to Adapt.
Model Updates in Embedded Deployment

Automated C-Code Generation

Embedded Systems

CPU

GPU

FPGA

MATLAB

C code

```c
10 static void classifyOnesphere(const real_T x[11994], cell_wrap_0 label
11 { real_T to Alpha(90); real_T expl_temp[6];
```
Model Updates in Embedded Deployment

Production Data

Embedded Systems

MATLAB

Update Parameters only
Agenda

Deploying AI to Embedded and Enterprise systems is difficult

Three specific challenges:
1. Limitations of Embedded hardware
2. Ongoing changes environment or system behavior
3. Scale to production load in Enterprise systems
Enterprise Deployment of AI
Integrate with Enterprise Systems and Scale to Production Load
Example: Incremental Health Monitoring

Sensor data

Anomaly Detection loop

```
while seqn % ... there's more data to process

    % Retrieve buffer of data
    datafilter = (sensordata.key == thisAsset) & (sensordata.SequenceNumber <= seqn+batchsize);
    streamdata = sensordata(datafilter, :);

    % Detect Anomalies with incremental One-class SVM
    [nextState, results] = detectAnomalyLocal(streamdata, state);

    % Remember results and update state of incremental learner
    anomalies(datafilter) = results.anomaly;
    score(datafilter) = results.score;
    timestamps(datafilter) = results.timestamp;
    state = nextState;

    seqn = seqn + batchsize; % step through batch test data

end
```
Incremental Learning within Streaming Architecture

\[
\text{incMdl} = \text{incrementalLearner}(\text{mdl});
\]

\[
\text{while dataStreaming}
\]

\[
\text{featureChunk} = \text{extractFeatures}(\text{streamdata});
\]

\[
\text{inclMdl} = \text{updateMetricsAndFit}(\text{incMdl}, \text{featureChunk}, \text{labels});
\]

\[
\text{End}
\]
Incremental Learning within Streaming Architecture
Operationalize AI without recoding
Model DevOps: Operationalize AI without recoding
Deploying AI to Embedded and Enterprise systems is difficult

Three specific challenges:
1. Limitations of Embedded hardware
2. Ongoing changes environment or system behavior
3. Scale to production load in Enterprise systems
Conclusions

Deploy to Embedded and Enterprise systems from one codebase

Tools for handling deployment-specific challenges:
- Fit models to embedded hardware with Quantization / Fixed-Point conversion
- Scale to data and users with MATLAB Production Server
- Incrementally adapt deployed models to maintain performance

Design, Deploy and Maintain AI-powered systems in one framework
Learn More

Check out our handout with links to customer stories, documentation – and examples which you can try out in MATLAB Online

DevOps for Software and Systems: Operationalization of Algorithms and Models

Deploying AI on PLCs
감사합니다