ALERT: Accurate Learning for Energy and Timeliness

Chengcheng Wan, Muhammad Husni Santriaji, Eri Rogers, Henry Hoffmann, Michael Maire and Shan Lu
DNN is Deployed Everywhere

Auto driving

Weather forecast

QA robot

Smart city

Trading

Text generator
DNN Deployment is Challenging.

Challenges

- Configuration space is huge
- Environment may change dynamically
- Must be low overhead
Previous Work

Previous Works

Challenges

Huge Space of Configuration

Dynamic Environment

Low Overhead

Resource Management

DNN design


...
Our ALERT System

Challenges

- Configuration space is huge
- Environment may change dynamically
- Must be low overhead
Our ALERT System

DNN & Power Cap Selection

Feedback-based estimation

Measurement

Challenges
- Configuration space is huge
- Environment may change dynamically
- Must be low overhead

Road
Evaluation Highlights

✔ ALERT satisfies LAE constraints.

99.9% cases for vision; 98.5% cases for NLP

✔ Probabilistic design overcomes dynamic variability efficiently.

ALERT achieves 93-99% of Oracle’s performance

✔ Coordinating App- and Sys-level improves performance.

Reduces 13% energy and 27% error over prior approach
Outline

- Understanding DNN Deployment Challenges
- ALERT Run-time Inference Management
- Experiments and Results
Outline

- Understanding DNN Deployment Challenges
- ALERT Run-time Inference Management
- Experiments and Results
Experiment Settings

**Tasks**
- Image classification (ImageNet)
- Sentence prediction (PTB)
- Question Answering (SQuAD)

**DNNs**
- ResNet50
- VGG16
- RNN
- Bert

**Platforms**
- ODroid
- CPUs
- GPU
Tradeoffs from DNNs

42 DNNs on ImageNet classifications

Inference Time of One Image (s)

Top-5 Error Rate (%)

MobileNet-v1 (α=1)
MobileNet-v2 (α=1.3)
ResNet50
NasNet-large
PnasNet-large

High accuracy comes with long latency.
Tradeoffs from System Settings

No setting is optimal for both energy and latency.
Run-time Variability

Without co-locate job

With co-locate job
Run-time Variability

Latency variation increased by co-located jobs.

Without co-locate job  With co-locate job
Potential Solutions

Combining both level achieves best performance.

Constraint Settings (deadline × accuracy_goal)
Outline

- Understanding DNN Deployment Challenges
- ALERT Run-time Inference Management
- Experiments and Results
Three Dimensions & Two Tasks

Maximize Accuracy
With energy consumption goal and inference deadline

Minimize Energy
With accuracy goal and inference deadline
Maximize Accuracy Task

Configurations

Constraints

Optimization

DNNs

Power cap

\[
\begin{array}{ccc}
1,1 & 1,2 & 1,3 \\
2,1 & 2,2 & 2,3 \\
3,1 & 3,2 & 3,3 \\
4,1 & 4,2 & 4,3 \\
\end{array}
\]

\[
\begin{align*}
\text{max}(A) \\
L & \leq X \\
E & \leq Y \\
\end{align*}
\]
How to estimate the inference latency?

- Two key challenges
  - Runtime variation: The inference time may be different even for same the configuration

<table>
<thead>
<tr>
<th>Profiling</th>
<th>50</th>
</tr>
</thead>
<tbody>
<tr>
<td>Runtime</td>
<td>52</td>
</tr>
</tbody>
</table>
How to estimate the inference latency?

- Two key challenges
  - Runtime variation
  - Too many combinations of DNNs and resources
Potential Solution

- Kalman filter
  - Estimate latency for each configuration
  - Use recent execution history

DNN2, P1: 52
DNN1, P2: 29

43  58  49

51

29  31

30

History  Prediction
Potential Solution: drawback

- Cannot solve the problem
  - Not enough history for each configuration
How to estimate the inference latency?

- Global Slow-down factor $\xi$
  - Use recent execution history under any DNN or resources

<table>
<thead>
<tr>
<th>DNN1, P1</th>
<th>Profiling</th>
<th>Runtime</th>
</tr>
</thead>
<tbody>
<tr>
<td>40</td>
<td>60</td>
<td></td>
</tr>
<tr>
<td>DNN2, P1</td>
<td>34</td>
<td>51</td>
</tr>
<tr>
<td>DNN1, P2</td>
<td>20</td>
<td>30</td>
</tr>
<tr>
<td>DNN2, P2</td>
<td>30</td>
<td>45</td>
</tr>
</tbody>
</table>

$\xi = 150\%$
How to estimate the inference latency?

- Mean estimation is not sufficient
  - The variation might be too big to provide a good prediction.
- Different implications on DNN selection

<table>
<thead>
<tr>
<th>Sequence 1</th>
<th>52</th>
<th>43</th>
<th>58</th>
<th>49</th>
<th>Mean 50</th>
<th>Variation 5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sequence 2</td>
<td>51</td>
<td>50</td>
<td>49</td>
<td>49</td>
<td>Mean 50</td>
<td>Variation 1</td>
</tr>
<tr>
<td>Sequence 3</td>
<td>15</td>
<td>99</td>
<td>10</td>
<td>70</td>
<td>Mean 50</td>
<td>Variation 40</td>
</tr>
</tbody>
</table>
How to estimate the inference latency?

- Global Slow-down factor $\xi$
  - Use recent execution history under any DNN or resources
  - Estimate its distribution: mean and variance

<table>
<thead>
<tr>
<th>History</th>
<th>Mean</th>
<th>Variation</th>
</tr>
</thead>
<tbody>
<tr>
<td>52</td>
<td>50</td>
<td>5</td>
</tr>
<tr>
<td>43</td>
<td></td>
<td></td>
</tr>
<tr>
<td>58</td>
<td></td>
<td></td>
</tr>
<tr>
<td>49</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
How to estimate accuracy under a deadline?

- Can inference be finished before deadline?
  - If yes, training accuracy of the selected DNN
  - If not, random guess accuracy
    - Unless it’s an Anytime DNN.
What is an Anytime DNN?

How to estimate accuracy under a deadline?

- Can inference be finished before deadline?
  - If yes, training accuracy of the selected DNN
  - If not,
    - Traditional DNN: random guess accuracy.
    - Anytime DNN: accuracy of the last output
How to estimate accuracy under a deadline?

Accuracy-Latency Relation + Latency Distribution = Expectation of Accuracy
How to manage energy?

- Power-cap as a knob to configure system resource
- Idle power: other process may still consume energy when DNN inference has finished

![Diagram showing energy management over time with Power-cap as a knob to configure system resource and different DNN states: DNN active1, DNN active2, and DNN Idle. The graph illustrates the latency target and time scale.](image-url)
How to estimate the energy consumption?

- Estimate energy from power
  - DNN active power is power setting
  - DNN idle power is estimated by Kalman filter
Our ALERT System

DNN & Power Cap Selection

Feedback-based estimation

Measurement

DNN System

Road
Outline

- Understanding DNN Deployment Challenges
- ALERT Run-time Inference Management
- Experiments and Results
Experiment Settings

Platforms
CPUs, GPU

DNNs
Sparse ResNet50, RNN

Tasks
1. Minimize energy
2. Maximize accuracy

Scenarios
Default, Compute intensive (2), Memory intensive (2)
Schemes

Oracles

• **Oracle**: Change configuration for every input. Assume perfect knowledge of future. Emulated from profiling result.
• **Oracle-static**: Same configuration for all inputs.

Baselines

• **Sys-only**: Only adjust power-cap
• **App-only**: Use an Anytime DNN
• **No-coord**: Anytime DNN without coordination with power-cap
Evaluation: Scheduler Performance

Average performance normalized to Oracle_Static (Smaller is better)
Evaluation: Scheduler Performance

Average performance normalized to Oracle_Static (Smaller is better)

- App-only
- Sys-only
- No-coord
- Sys+App(ALERT)
- Oracle

Violations (%)

Minimize Error
How ALERT Works with Traditional DNN

- Meet requirements in most cases
- Quickly detect contention changes
- Use anytime DNN under unstable environment
How ALERT Works with Traditional DNN

- Meet requirements in most cases
- Quickly detect contention changes
- Use anytime DNN under unstable environment
How ALERT Works with Anytime +Traditional DNN

- Meet requirements in most cases
- Quickly detect contention changes
- Use anytime DNN under unstable environment
Conclusion

- Understand DNN inference challenges
- ALERT Run-time inference System
- High performance and energy efficiency