Animal Recognition and Identification with Deep Convolutional Neural Networks for Farm Monitoring

Ryunosuke Murakami, Yuichi Okuyama, Abderazek Ben Abdallah
Graduate School of Computer Science and Engineering,
Adaptive Systems Laboratory
The University of Aizu, Japan
Outline

• Background
• Motivation
• Research goal
• Approach
• Evaluation
• Conclusion
Background

• Due to the increasing demand in the agricultural industry, the need to effectively grow and protect a plant and increase its yield is necessary.

• It is important to monitor the plant during its growth period and protect it from animals (pig, wolf, etc.) at the time of harvest.

Fig 1. Farm and wild animals

Fig 2. Olive and Date Tree Diseases
Motivation

• Monitoring the plants from plantation to harvesting is necessary for better productivity
• Smart farming needs right decision and monitoring tools for better productivity, quality and profit
• Artificial neural network concept is efficient for image processing
Convolutional Neural Network

- Efficient to object recognition
  - Process data while keeping the shape of image
- Behavior is similar to *visual cortex*
  - Performance is close to human-level
- Learn *feature vector* automatically from data

Fig 3. CNN example

- pig
- other
Research goal

• Develop an efficient FARM monitoring system for better productivity, quality and profit based on artificial neural network concept:
  – Hardware implementation of Deep Convolutional Neural Network on FPGA
  – Evaluation of real hardware complexity (power and area) and performance (recognition accuracy, time)

• The purpose is to monitor strange animals (pig, etc) and diseases on the stem/leaf/fruits of the crop
System overview

Fig 4. System overview: OASIS FMS-1
Flow of recognition system

- Region proposal
  1. Create initial region with pixel
  2. Group the similar regions
  3. Continue “2” until the whole image becomes a single region

Fig 5. Flow chart of recognizer

出典: 「Rich feature hierarchies for accurate object detection and semantic segmentation」
Approach

1. Software implementation by Python
   – Design D-CNN using Chainer framework

2. Implement selective-search and integrate
   – Region proposal for object position

3. Hardware implementation on FPGA by HDL
   – Install parameters already learned
My network structure

– Input image
  • Image size: 32x32
  • Channel depth=3 (RGB)

– Hidden size
  • 3 series of conv-layer
    (conv + Pooling)

– Output layer
  • ReLU
  • Dropout
  • Fully connect

Fig 5. network structure
Dataset

• Collected from *ImageNet*
• Image: 32x32 pixel, channel = 3(RGB)
• Distribution of original data
  – Pig images: 685
  – Other images: 800

Table 1. Data distribution

<table>
<thead>
<tr>
<th>Class</th>
<th>Train</th>
<th>Valid</th>
<th>Test</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pig</td>
<td>549</td>
<td>68</td>
<td>68</td>
</tr>
<tr>
<td>Other animals</td>
<td>664</td>
<td>68</td>
<td>68</td>
</tr>
</tbody>
</table>
Data augmentation

• Augmented training data twice
  – Original data(Training): 1,213
  – Augmented data: 2,426

[applied image conversion]
1. Slide the pixels randomly within range (-4 ~ 4)
2. Fill in “0” with empty space
3. Flip horizontal randomly
Learning methods

• In the following experiments, 3 learning methods were compared

Table 2. Learning methods

<table>
<thead>
<tr>
<th>Learning method</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>Momentum SGD</td>
<td>Optimized by gradient of loss function and learning rate is decreased with gradient</td>
</tr>
<tr>
<td>AdaGrad</td>
<td>Learning rate decreases with scale of weights</td>
</tr>
<tr>
<td>Adam</td>
<td>Combined with Momentum and AdaGrad</td>
</tr>
</tbody>
</table>
Evaluation configurations

• Learning parameters
  – Batch size: 128
  – Iterate num: 100
  – Learning decay: 0.1 times in each 20 epoch

• Experiment environment

<table>
<thead>
<tr>
<th>Table 3. Machine spec</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>OS</strong></td>
</tr>
<tr>
<td><strong>CPU</strong></td>
</tr>
<tr>
<td><strong>GPU</strong></td>
</tr>
<tr>
<td><strong>Language</strong></td>
</tr>
<tr>
<td><strong>Library</strong></td>
</tr>
</tbody>
</table>
Evaluation result - Best Model

[Distribution rate]
Data(1221) = {pig: 549, oth: 664}

[Batch size]
mini_batch(128) = {pig: 64, oth: 64}

[Network structure]
Conv1(filter=16, stride=1, pad=2)
Conv2(filter=32, stride=1, pad=2)
Conv3(filter=32, stride=1, pad=2)
l1=L.Linear(4 * 4 * 32, 1000),
l2=L.Linear(1000, 2)

[Learning parameters]
--optimizer: Adam
--iter 100
--lr_decay_iter 20
--Activation: ReLU

Best test accuracy: 92.647

Fig 6. Learning result
Accuracy evaluation with different filter numbers

Table 4. Evaluation with network

<table>
<thead>
<tr>
<th>Conv1</th>
<th>Conv2</th>
<th>Conv3</th>
<th>Accuracy</th>
<th>Elapsed time</th>
<th>Error</th>
<th>Elapsed time</th>
</tr>
</thead>
<tbody>
<tr>
<td>64</td>
<td>64</td>
<td>64</td>
<td>7.353</td>
<td>0.184</td>
<td></td>
<td></td>
</tr>
<tr>
<td>32</td>
<td>64</td>
<td>64</td>
<td>9.559</td>
<td>0.136</td>
<td></td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>64</td>
<td>64</td>
<td>10.294</td>
<td>0.117</td>
<td></td>
<td></td>
</tr>
<tr>
<td>64</td>
<td>32</td>
<td>64</td>
<td>9.559</td>
<td>0.154</td>
<td></td>
<td></td>
</tr>
<tr>
<td>32</td>
<td>32</td>
<td>64</td>
<td>8.824</td>
<td>0.114</td>
<td></td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>32</td>
<td>64</td>
<td>11.765</td>
<td>0.104</td>
<td></td>
<td></td>
</tr>
<tr>
<td>64</td>
<td>64</td>
<td>32</td>
<td>8.088</td>
<td>0.178</td>
<td></td>
<td></td>
</tr>
<tr>
<td>32</td>
<td>64</td>
<td>32</td>
<td>8.088</td>
<td>0.128</td>
<td></td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>64</td>
<td>32</td>
<td>9.559</td>
<td>0.109</td>
<td></td>
<td></td>
</tr>
<tr>
<td>64</td>
<td>32</td>
<td>32</td>
<td>9.559</td>
<td>0.148</td>
<td></td>
<td></td>
</tr>
<tr>
<td>32</td>
<td>32</td>
<td>32</td>
<td>10.294</td>
<td>0.11</td>
<td></td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>32</td>
<td>32</td>
<td>7.353</td>
<td>0.1</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- Conv1,2,3: each convolutional layer
- Error = 100 * (1 – accuracy)
- Accuracy: ratio of concordance with correct label
- Elapsed time(μs) is calculated to batch data(128 images) classification
### Table 5. Error rate evaluation with different learning method

<table>
<thead>
<tr>
<th>Network (filter number)</th>
<th>momentum-SGD (error/time)</th>
<th>Adam (error/time)</th>
<th>adaGrad (error/time)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(16, 32, 32)</td>
<td>7.353/0.1</td>
<td>9.559/0.097</td>
<td>12.5/0.097</td>
</tr>
<tr>
<td>(32, 32, 64)</td>
<td>8.824/0.114</td>
<td>9.559/0.115</td>
<td>19.118/0.114</td>
</tr>
<tr>
<td>(32, 64, 32)</td>
<td>8.088/0.128</td>
<td>8.823/0.125</td>
<td>13.235/0.128</td>
</tr>
</tbody>
</table>

(Time: μs)

### Table 6. Error rate evaluation with different Activation function

<table>
<thead>
<tr>
<th>network</th>
<th>ReLu</th>
<th>sigmoid</th>
<th>tanh</th>
</tr>
</thead>
<tbody>
<tr>
<td>(16,32,32)</td>
<td>7.353/0.1</td>
<td>11.029/0.099</td>
<td>8.088/0.101</td>
</tr>
<tr>
<td>(32,32,64)</td>
<td>8.824/0.114</td>
<td>11.765/0.114</td>
<td>8.824/0.112</td>
</tr>
<tr>
<td>(32,64,32)</td>
<td>8.088/0.128</td>
<td>8.824/0.127</td>
<td>9.559/0.127</td>
</tr>
</tbody>
</table>

(Time: μs)
Conclusions and future work

• This paper presented an Animal Recognition and Identification with Deep Convolutional Neural Networks for Farm Monitoring

• As a first step, the system was designed and evaluated in software

• Evaluation results shows that the system achieves 92.647 accuracy and 0.1μs
Conclusions and future work

• As a future work, we intend to design the system in hardware (Verilog and FPGA) and evaluate its real performance, complexity, and power consumption
Thank you for your kind attention.