Search Optimization for JPEG Quantization Tables
using a Decision Tree Learning Approach

Sharon Gieske
6167667

Supervisors: Zeno Geradts (NFI)
Master System and Network Engineering
University of Amsterdam

2014-07-02
Table of Contents

Introduction
  Motivation
  Decision tree learning algorithm
  Research Question

Approach
  Overview
  Data Preprocessing and Training
  Evaluation

Results

Conclusion

Questions
Motivation

- Growing popularity for taking pictures
- Digital images often recovered in forensic investigations
- Identify origin of images to a specific camera or common source
- Large sets of images are retrieved

**Camera Identification:**

- Intrinsic features of camera hardware give more reliable results[2]
- Sensor Imperfections, CFA Interpolation, Image Features
JPEG quantization tables

JPEG compression:
- RGB to Luminance-Chrominance colour space
- Splitting into two $8 \times 8$ blocks
- Discrete Cosine Transform (spatial domain $\rightarrow$ frequency domain)
- Compression ratio
- Correlated to camera make/model

‘..is reasonably effective at narrowing the source of an image to a single camera make and model or to a small set of possible cameras.’[1]
Decision tree learning algorithm

Camera identification problem → pattern recognition problem:
  ▶ map feature set to corresponding label

Decision tree learning algorithm:
  ▶ Rule based, generates best splits
  ▶ Simple to interpret / human readable
Can searching through JPEG quantization tables be optimized with the use of decision tree learning?

Subquestions:

1. Can identifiable parameters be found in JPEG quantization tables?
2. What is the performance of decision tree learning with JPEG quantization tables?
Overview

1. Extract quantization tables from images
2. Generate feature set
3. Train decision tree classifier (make/model)
4. Evaluate classifications
5. Compare against method using hash database
Data Preprocessing and Training

1. Extract quantization tables from images
   - Unix command: djpeg

2. Generate feature set
   - Add features: sum, min, max, mean, median, var, std
   - Run feature selection

3. Train decision tree classifier
   - CART: combines classification and regression trees
Evaluation

4. Evaluate with weighted $F_\beta$-score
   - Recall is more important: $\beta = 2$
   \[
   F_\beta = 1 + \beta^2 \ast \frac{\text{precision} \ast \text{recall}}{(\beta^2 \ast \text{precision}) + \text{recall}}
   \] (1)

5. Compare against method using hash database
   - Database of hashed quantization tables
     - 1→1 mapping
     - 1→n mapping
   - Use same training and validation data
Results

Dataset:
- 45,666 images (NFI & Dresden Image Database)
- 41 camera models
- 19 camera makes
- 1,016 unique quantization tables

Identifiable parameters: 50 out of 128
603 nodes, depth of 26

Figure: Partial Decision Tree
## Zoom in: F2-score for camera make

<table>
<thead>
<tr>
<th>Make</th>
<th>F2</th>
<th>Make</th>
<th>F2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Kodak</td>
<td>99 %</td>
<td>Praktica</td>
<td>43 %</td>
</tr>
<tr>
<td>Ricoh</td>
<td>94 %</td>
<td>Nikon</td>
<td>86 %</td>
</tr>
<tr>
<td>Panasonic</td>
<td>79 %</td>
<td>Casio</td>
<td>99 %</td>
</tr>
<tr>
<td>PS</td>
<td>100%</td>
<td>Canon</td>
<td>98 %</td>
</tr>
<tr>
<td>Olympus</td>
<td>64 %</td>
<td>Logitech</td>
<td>100%</td>
</tr>
<tr>
<td>Sony</td>
<td>58 %</td>
<td>Motorola</td>
<td>100%</td>
</tr>
<tr>
<td>Agfa</td>
<td>78 %</td>
<td>Epson</td>
<td>100%</td>
</tr>
<tr>
<td>Rollei</td>
<td>84 %</td>
<td>BlackBerry</td>
<td>100%</td>
</tr>
<tr>
<td>Samsung</td>
<td>67 %</td>
<td>Pentax</td>
<td>80 %</td>
</tr>
<tr>
<td>FujiFilm</td>
<td>96 %</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Table:** F2-score for camera make
Decision tree vs Hash databases

- 5-Fold Stratified Cross Validation
- 80 % Train set, 20 % Validation set

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Precision</th>
<th>Recall</th>
<th>F2-score</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hash (1-1)</td>
<td>79 %</td>
<td>68 %</td>
<td>68 %</td>
</tr>
<tr>
<td>Hash (1-n)</td>
<td>50 %</td>
<td>99 %</td>
<td>83 %</td>
</tr>
<tr>
<td>Decision tree</td>
<td>90 %</td>
<td>89 %</td>
<td>89 %</td>
</tr>
</tbody>
</table>

**Table: Camera Make Identification**

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Precision</th>
<th>Recall</th>
<th>F2-score</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hash (1-1)</td>
<td>54 %</td>
<td>39 %</td>
<td>37 %</td>
</tr>
<tr>
<td>Hash (1-n)</td>
<td>50 %</td>
<td>98 %</td>
<td>83 %</td>
</tr>
<tr>
<td>Decision tree</td>
<td>78 %</td>
<td>82 %</td>
<td>80 %</td>
</tr>
</tbody>
</table>

**Table: Camera Model Identification**
Discussion

- Both methods are prone for overfitting
- Hash database holds larger search space
- Training hash database is quicker
Conclusions

- Parameters can be reduced to 50
- Decision tree classifier gains better F2-score of 89% (make)
- 1→N hash database gains better F2-score of 83% (model)

- Decision tree classifier is more flexible, reduces search space, but harder to train than 1→N hash database

Future work:
- Compare to other learning algorithms
  - Naive Bayes
- Extend feature set
Questions?
References I
